Sterling Selling and Fulfillment Foundation

条件ビルダーの拡張

バージョン 9.1

Sterling Selling and Fulfillment Foundation

条件ビルダーの拡張

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、 35 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 Condition Builder
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 条件ビルダー・フィールドのカスタマイズ

条件ビルダーについて

Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales 条件ビルダーは、サービス定義フレームワークの一部としてまたはパイプラインの定義で使用されます。これは、動的条件を定義するが、条件ビルダーにある属性の静的セットの一部も使用する場合に使用できます。

注: 条件ビルダーで行われるストリング比較は、すべて大/小文字を区別します。

サービスに対する条件の定義と条件ビルダーの使用について詳しくは、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales: *構成ガイド*を参照してください。

プロセス・タイプごとのカスタム属性の追加

このタスクについて

オーダー・フルフィルメント、出荷などのプロセス・タイプに基づいて、カスタム属性をステートメント・ビルダーに追加できます。このカスタマイズは、条件ビルダーで定義済みのセットとして提供されていない属性に基づいて条件を評価する場合に便利です。これらのカスタム属性は、条件作成時に条件ビルダーに追加されます。条件の作成について詳しくは、「この世で人々があなたについて話したことよりも厄介なことがあるとしたら、それは誰もあなたについて話をしないことです」を参照してください。Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales: 構成ガイド

カスタム属性は、以下のようにカスタム XML ファイルを作成することによって、追加できます。

手順

1. *INSTALL_DIR*/extensions/global/template/configapi ディレクトリーに拡張の名前を付けたディレクトリーを作成します。
2. *INSTALL_DIR*/extensions/global/template/configapi ディレクトリーに extn_conditionbuilder.xml という名前のファイルを作成します。サンプル XML ファイルの形式を以下の例に示します。

   注: オーダー保留タイプの条件ビルダーをカスタマイズするには、holdtype_extn_conditionbuilder.xml というファイルを作成し、このセクションで説明する同じ手順に従います。

   注: アプリケーション・マネージャー画面に変更を表示するには、extn_conditionbuilder.xml ファイルに条件を追加するたびにアプリケーション・サーバーを再始動する必要があります。

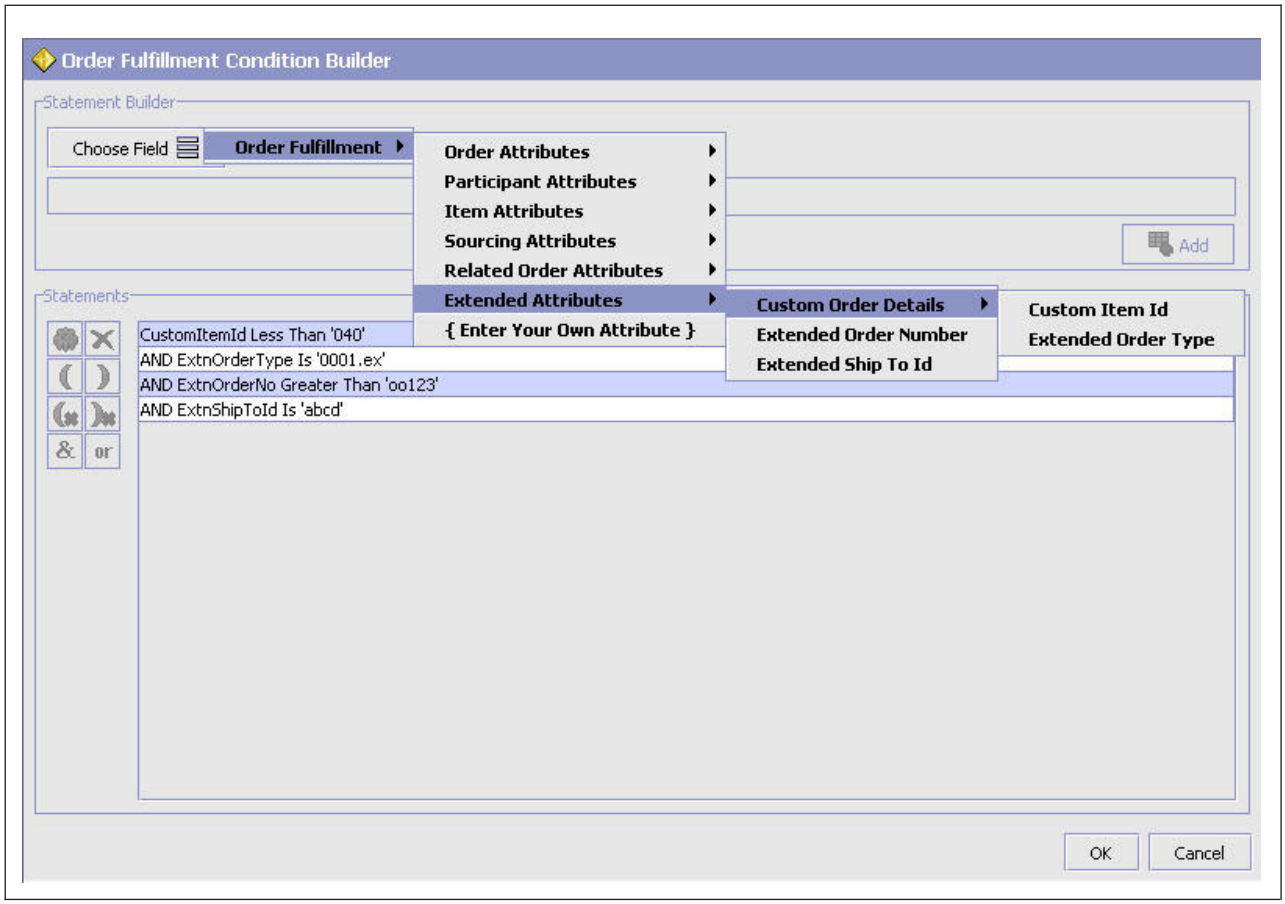
サンプル・ファイルは、拡張オーダー番号、出荷情報およびオーダーの詳細に関する条件を作成します。拡張オーダーの詳細には、CustomItemId や ExtnOrderType などのサブ要素があります。

ID 属性は、要素名と同じである必要があります。例えば、要素 ExtnOrderNo には、属性 Id="ExtnOrderNo" が含まれている必要があります。

```
<CustomConditionAttributes>
   <ProcessType Name="ORDER_FULFILLMENT">
       <ExtnOrderNo Id="ExtnOrderNo" DisplayName="Extended Order Number"
        DataType="Text-40" Type="Leaf"/>
       <ExtnShiptoId Id="ExtnShiptoId" DisplayName="Extended Ship To Id"
        DataType="Text-40" Type="Leaf"/>
       <CustomOrderDetails Id="CustomOrderDetails"
        DisplayName="Custom Order Details">
          <CustomItemId Id="CustomItemId" DisplayName="Custom Item Id"
            DataType="Text-100" Type="Leaf"/>
          <ExtnOrderType Id="ExtnOrderType" DisplayName="Extended Order Type"
           DataType="Text-100" Type="Leaf"/>
        </CustomOrderDetails>
   </ProcessType>
</CustomConditionAttributes>
```

属性	説明
プロセス・タイプ	
Name	必須。条件を作成するプロセス・タイプの名前を指定します。
要素	
DisplayName	必須。条件ビルダー画面でのこの属性の表示名を指定します。
DataType	必須。この属性に必要なデータ・タイプを指定します。例えば、定義済み属性 BillToId のデータ・タイプは「 ID-40 」です。 データ・タイプは、*INSTALL_DIR*/repository/datatypes/datatypes.xml ファイルで提供されるタイプのいずれかにできます。
ID	必須。この条件を関連付ける ID を指定します。デフォルトでは、これは要素名と同じです。
型	必須。この属性のタイプを指定します。例えば、「Leaf」タイプは、メニュー・アイテムの一部として属性を指定します。 これを指定しないと、この属性は、サブメニューに導かれるメニュー・ヘッダーとして見なされます。この例では、要素 CustomOrderDetails に CustomItemId や ExtnOrderType のような先導サブメニューがあるため、この要素のタイプは定義されていません。

3. この例で示されるような ExtnOrderNo、ExtnShipToId などの要素は、データベースに保存されるときに条件を定義するために使用されます。したがって、要素名は、単一プロセス・タイプに対して固有である必要があります。
4. この XML ファイルの内容は、プロセス・タイプごとに定義済みの静的属性とマージされ、以下のように画面に表示されます。



5. ただし、「ステートメント・ビルダー」ウィンドウで「**OK**」をクリックすると、XML の表示名が「条件の詳細」ウィンドウに表示されます。
6. この表示名は、条件が保存されると、これらの要素名に置き換えられます。これらの要素名およびその条件は、データベース表 YFS_CONDITION で条件値と呼ばれます。
7. これらのカスタム属性は、条件ビルダーで現在の静的属性と似た方法で表示されます。

   このように定義されたカスタム属性は、サービス定義フレームワークの一部として、またはイベントあるいは条件を作成するパイプラインの定義で使用できます。ただし、パイプラインの定義で使用されると、この条件エバリュエーターは、これらのカスタム属性を評価しません。

   注: カスタム属性は、パイプラインの決定ルールの条件の評価には使用できません。

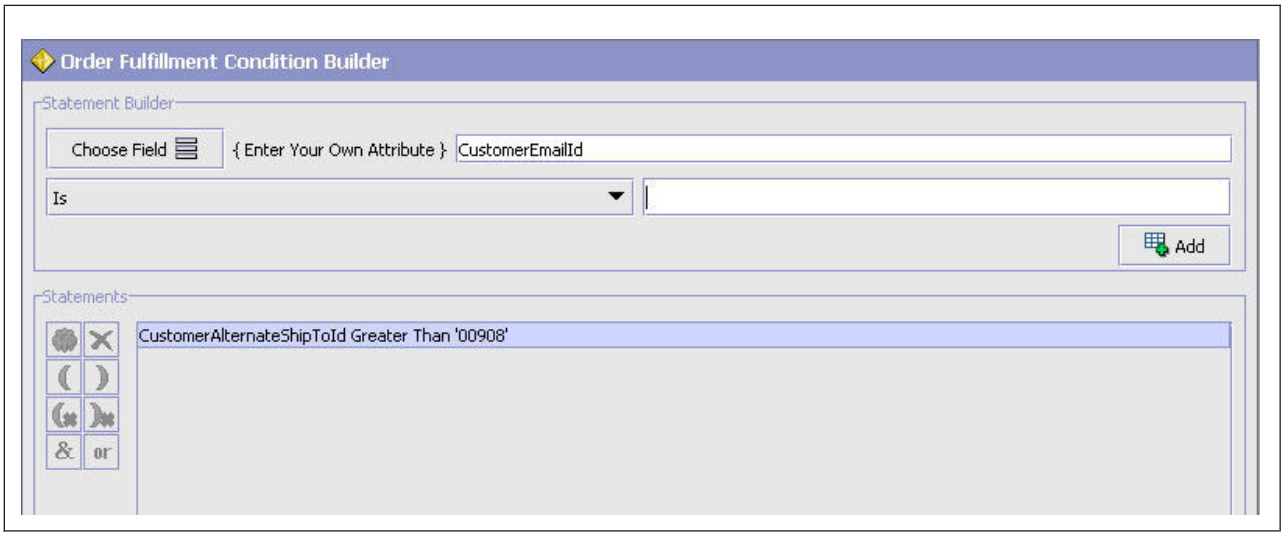
条件定義中のカスタム属性の追加

このタスクについて

特定のオーダーに対して条件を定義する場合、条件の一部として評価される独自のカスタム属性を追加することをオプションで選択できます。

手順

1. 「ステートメント・ビルダー」ウィンドウで、「**フィールドの選択**」 をクリックして「**自分の属性を入力してください**」オプションを選択することによって、カスタム属性を条件ビルダーで追加できます。
2. 以下に示すように条件に含める属性名を入力し、「**追加**」をクリックします。



例えば、CustomerEmailId という属性を含める場合、その属性を指定されたテキスト・ボックスに入力し、ステートメントの作成を継続します。この属性は、条件の評価に組み込まれます。

注: 条件に対して対応するテンプレートを拡張し、上記の属性を含める必要があります。

このフィールドは、入力できる任意の XML 属性とは逆に、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales によって定義済みの非公開のキー属性に限定されます。

3. 名前を入力すると、定義済みの属性と同じ方法で残りの条件をビルドできます。定義済みの属性を使用して条件を作成する方法について詳しくは、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales: 構成ガイドを参照してください。

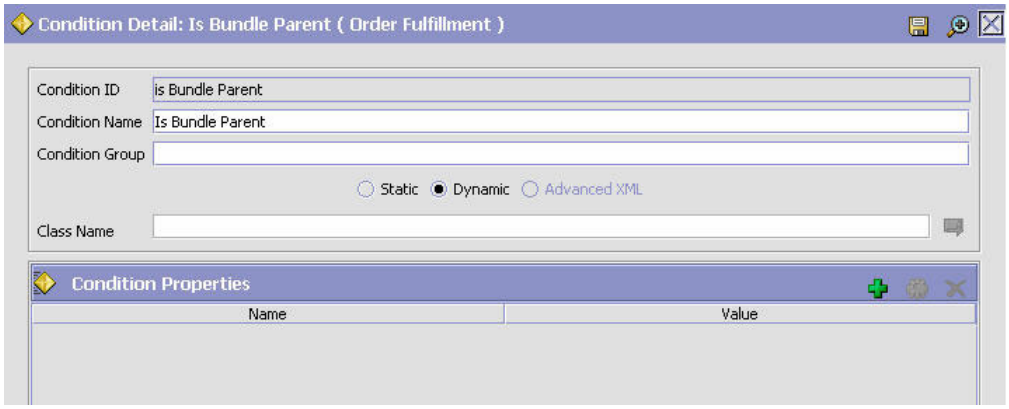
   条件定義中に作成されたカスタム属性は、サービス定義フレームワークの一部として、またはイベントあるいは条件を作成するパイプラインの定義で使用できます。

   注: この方法で定義したカスタム属性は、この条件を定義するときのみ使用できます。入力された属性は、他の条件では再使用できません。

動的条件の条件のプロパティーの指定

条件定義では、静的条件、動的条件または拡張 XML 条件を作成できます。

動的条件を作成するとき、「動的」フィールドを確認し、条件を評価するために呼び出すクラス名を指定する必要があります。この動的条件の作成は、拡張され、カスタム名、値プロパティーの構成を含みます。これらのプロパティーは、条件を評価する前に Java クラスに設定されます。



これらの拡張属性を有効にするには、動的条件クラスは YCPDynamicConditionEx インターフェースを実装する必要があります。このインターフェースの定義を以下に示します。

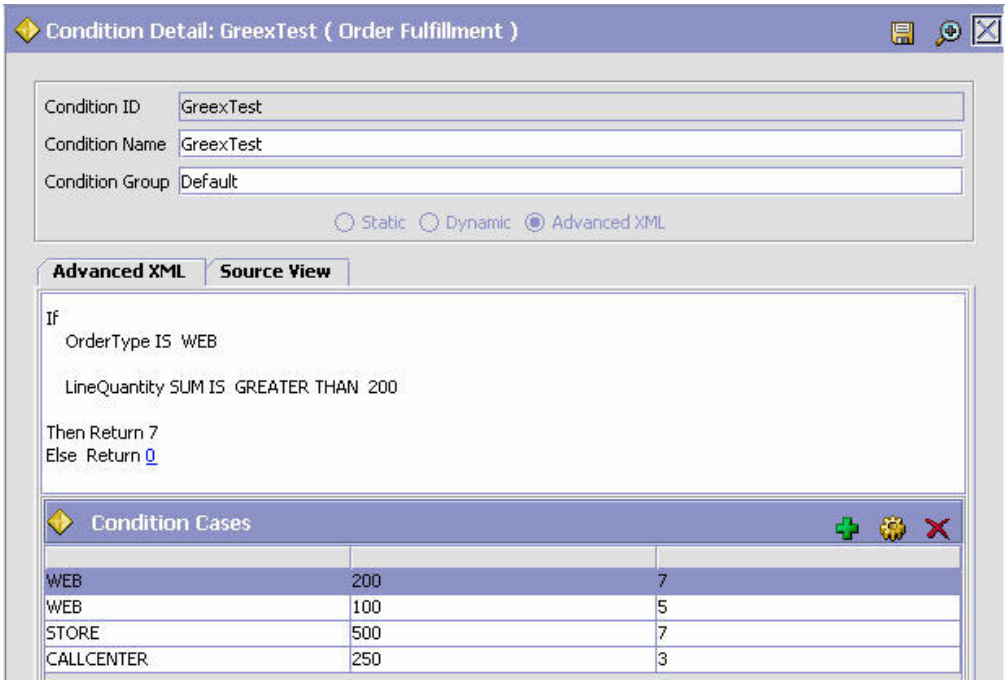
```
public interface YCPDnamicConditionEx
{
  boolean evaluateCondition(YFSEnvironment env, String name, Map mapData,
    Document doc);
  void setProperties(Map map);
}
```

インターフェースに渡されるパラメーター名は、定義中に構成された条件名です。

インターフェースおよびパラメーターについて詳しくは、Javadocs を参照してください。

拡張 XML 条件の条件のケースの指定

決定表ベースの拡張 XML 条件を変更している場合、拡張 XML 条件を拡張して、カスタム・ケースのさまざまな属性を含めることができます。これらの属性は、決定表ベースの拡張 XML 条件のタイプを評価する前に設定されます。



決定表ベースの拡張 XML 条件に定義された既存のケースを追加または変更できます。条件のケースのさまざまな属性を変更できます。条件のケースを変更し、「**保存**」ボタンをクリックした後、属性の新しい値は「拡張 XML」画面と「ソース・ビュー」画面の両方に反映されます。

決定表ベースの拡張 XML 条件に対して定義されたデフォルトの戻り値は、「拡張 XML」画面にハイパーリンクとして表示されます。満たされる条件のケースがない場合、Greex エンジンはデフォルト値を返します。ハイパーリンクをクリックして、デフォルトの戻り値を編集し、拡張 XML 条件に対して新しいデフォルトの戻り値を指定します。ポップアップ画面に古い値が表示されます。新しい値を入力することもできます。ポップアップ画面で「**保存**」ボタンをクリックすると、新しい値が「拡張 XML」画面および「ソース・ビュー」画面で反映されます。

第 3 章 拡張 XML 条件の作成および変更

拡張 XML 条件について

拡張 XML 条件は、Greex ルールとも呼ばれています。新しい値を拡張 XML 条件の変更可能パラメーターに割り当てることができるだけです。拡張 XML 条件または Greex ルールに関する情報をローカライズまたはログに記録することも可能です。

Greex とは

Greex フレームワークを使用すると、Greex 構文に基づいて拡張 XML 条件を定義できます。Greex フレームワークは、拡張 XML 条件を評価するために、入力 XML 文書を使用する宣言メカニズムを提供します。

Greex フレームワークの特徴的な機能を以下に示します。

- XML 対応です。
- 名前空間をサポートします。
- ライブラリーのセットが含まれています。
- XML 要素、ストリング、またはブール値を返すことができます。

Greex ライブラリー関数

Greex フレームワークには、XPath 式を使用して入力データで呼び出すことができる関数のセットから構成される Greex ライブラリーが含まれています。関数仮パラメーターに定数を使用することもできます。Greex ライブラリーには、以下の定義済みの関数のセットが用意されています。

- booleanAnd — Xpath 属性リストのすべての値が TRUE の場合、TRUE を返します。
- booleanOr — Xpath 属性リストのいずれかの値が TRUE の場合、TRUE を返します。
- compareAll — コレクションのすべての値を比較します。
- compareAny — コレクションのいずれかの値を比較します。
- count — 指定した Xpath の発生回数をカウントします。
- dateAdd — 指定した日数を日付に加算します。
- dateGreater — 2 つの日付を比較します。
- dateGreaterOrEqual — 2 つの日付を比較します。
- dateMax — Xpath 要素の日付のセットから最大の日付を返します。
- dateMin — Xpath 要素の日付のセットから最小の日付を返します。
- doubleAdd — 2 つの 2 倍の値を加算します。
- doubleGreaterOrEqual — 2 つの 2 倍の値を比較します。
- doubleLesser — 2 つの 2 倍の値を比較します。

- doubleLesserOrEqual — 2 つの 2 倍の値を比較します。
- doubleMax — Xpath 要素の最大の属性値を 2 倍として返します。
- doubleMin — Xpath 要素の最小の属性値を 2 倍として返します。
- doubleSum — Xpath 要素の属性値の合計を 2 倍として返します。
- equals — 同じタイプの 2 つのオブジェクトを比較します。
- equalsIgnoreCase — 2 つのストリングを比較します。大/小文字は無視されます。
- intAdd — 2 つの整数値を加算します。
- intGreater — 2 つの整数値を比較します。
- intGreaterOrEqual — 2 つの整数値を比較します。
- intMax — Xpath 要素の最大の属性値を整数として返します。
- intMin — Xpath 要素の最小の属性値を整数として返します。
- intSum — Xpath 要素の属性値の合計を整数として返します。
- isTrue — パラメーター値が `true` に設定されているかどうか確認します。
- isVoid — 渡されたパラメーターが NULL または空かどうか確認します。
- listIntersection — Xpath 要素の属性値 (つまり、コンマ区切り値) の論理積をストリングとして返します。
- listUnion — Xpath 要素の属性値 (つまり、コンマ区切り値) の和集合をストリングとして返します。
- stringBegins — 1 番目のストリングが 2 番目のストリングで始まっているかどうか確認します。

Greex 構文とは

Greex 構文を使用して、拡張 XML 条件または Greex ルールを作成できます。拡張 XML 条件は、入力データに関する特定の条件を評価するために使用されます。Greex 構文は XML ベースです。このスタイルの条件評価では、入力データをブール出力だけではなく、複数の方法で使用できます。Greex 構文では、複数の IF および ELSE のブロックを使用してネスト化できる Greex 構造が提供されます。また、AND 演算子または OR 演算子を使用したグループ式も使用できます。それぞれの式は、1 つ以上の関数呼び出しから構成されます。ネスト化されたループの形式で関数を含めることができます。つまり、関数のパラメーターを他の関数呼び出しにすることができます。これらには、基本的な IF ELSE 条件が含まれます。以下の表では、If/Else 構造のさまざまな要素について説明しています。

要素	説明
条件	「条件」要素は、入力 XML に対して命令することによって、評価する必要があるすべての式の論理グループを提供します。

要素	説明

返品	If/Else 構造で定義された各条件は、値を返す必要があります。条件は、XML 要素、ストリング、またはブール値を返すことができます。条件は、「返品」要素で If/Else 条件の適切な値を返します。
	例:
	• XML 文書を返す場合、「返品」要素を以下のようにできます。
	`<Return>`
	`<Value output="<Order`
	`type=&quot;Web&quot;`
	`discount=&quot;5&quot;/>"/>`
	`</Return>`
	• ストリングを返す場合、「返品」要素を以下のようにできます。
	`<Return>`
	`<Value output="Draft Order Created"/>`
	`</Return>`
	• ブール値を返す場合、「返品」要素を以下のようにできます。
	`<Return>`
	`<Value output="true"/>`
	`</Return>`
	• 評価の結果として「PureXML」要素を返す場合、「返品」要素を以下のようにできます。
	`<Return>`
	`<Value output="<Date"/>`
	`</Return>`
	上記の出力は、次のようになります。
	`<Date/>`

「条件」要素には、さまざまな「式 (Expression)」要素が含まれています。それぞれの「式 (Expression)」要素は、指定した条件に対して評価する式を定義します。以下の表では、条件のさまざまな要素について説明しています。

要素	説明

式 (Expression)	「式 (Expression)」要素には、満たすべき条件に対して評価する式が含まれています。式で関数呼び出しを行うには、関数名の先頭に「fn:」を付けます。例えば、関数呼び出しを行う以下の「式 (Expression)」要素を定義します。
	`<Expression>fn:equals(@ZipCode,"01876")</Expression>`
	関数をパラメーターとして他の関数に渡すこともできます。例えば、以下の「式 (Expression)」要素を定義して、関数をパラメーターとして他の関数に渡すことができます。
	`<Expression>fn:intGreater`
	`(fn:count(OrderLines/OrderLine), "100")</Expression>`

要素	説明
グループ	これはオプションの要素です。式のセットを一緒に評価する場合、その式のセットをグループ化する必要があります。 「グループ」要素の「op」属性は、式のセットに対して実行する命令を示します。有効な値は、「or」および「and」です。「グループ」要素では、複数の要素を定義できます。「グループ」要素の op 値が「and」の場合、「グループ」要素の一部であるすべての式の評価が「true」の場合のみ条件が満たされます。同様に、「グループ」要素の「op」属性が「or」の場合、「グループ」要素の一部である式のいずれかの評価が「true」の場合のみ条件が満たされます。 任意のレベルのネスト化された「グループ」および「式 (Expression)」の要素を作成できます。 注: 1 つの式を評価する場合、「グループ」要素を作成せずに、「条件」要素の下に「式 (Expression)」要素を定義してください。

拡張 XML 条件または Greex ルールを定義する IF ELSE 構造のサンプルを、次に示します。

```
<If>
  <Condition name="isWebOrder?">
    <Group op="and">
      <Expression>fn:!equals(@orderType,"WEB")</Expression>
      <Expression>fn:equals(address::@ZipCode, "01876")</Expression>
    </Group>
  </Condition>
  <Return>
     <Value output="&lt;Order type=&quot;Web&quot;
            discount=&quot;5&quot;/&gt;"/>
  </Return>
</If>
  <Else>
    <Return>
       <Value output="&lt;Order type=&quot;Catalog&quot;
              discount=&quot;2&quot;/&gt;"/>
     </Return>
  </Else>
```

カスタム Greex 関数の書き込み

このタスクについて

Greex フレームワークには、Greex ライブラリーの一部として、定義済みの関数のセットが用意されています。必要に応じて、カスタム Greex 関数を書き込むこともできます。

カスタム Greex 関数を書き込む手順は、次のとおりです。

手順

1. com.yantra.ycp.greex.library.LibraryFunction インターフェースのすべての関数を実装することにより、カスタム・ライブラリーの実装を提供します。以下の関数を実装する必要があります。
   - Object invoke(GreexContext ctx, List params)

- boolean validateParams(List params)
- String getName()
- String getDescription()
- String getReturnType()
- String[] getParamTypes()

2. LibraryFunctionFactory クラスの registerFunction (LibraryFunction 関数) メソッドを呼び出して、カスタム・ライブラリーを Greex フレームワークに登録します。

   例えば、MyCustomLibrary などのカスタム Java クラスを登録するには、以下の registerfunction() メソッドを呼び出します。

   ```
   LibraryFunctionFactory.getInstance().registerFunction(new MyCustomLibrary());
   ```

3. 手順 1 で作成したカスタム・ライブラリーの jar を作成し、これをアプリケーション・サーバーのクラスパスに追加します。
4. カスタム Greex の初期化指定子サーブレット (MyCustomGreexInitializer など) を作成して (通常のサーブレットでもかまいません)、カスタム・ライブラリーを Greex の LibraryFunctionFactory にロードします。以下のステップを実行します。
   a. *<INSTALL_DIR>*/repository/eardata/*<application_name>*smcfs/extn に web.xml.sample ファイルを生成する *<application_name_context>*smcfs.ear ファイルをビルドします。
   b. web.xml.sample の名前を web.xml に変更します。
   c. Greex フレームワークでカスタム・ライブラリー関数をロードするには、*<INSTALL_DIR>*/repository/eardata/*<application_name>*smcfs/extn/web.xml ファイルでサーブレットの名前を指定します。例えば、サーブレット・クラスが com.servlet.MyCustomGreexInitializer の場合、以下のエントリーを web.xml ファイルに追加します。
   d. *<application_name_context>*smcfs.ear ファイルを再ビルドします。

   ```
   <servlet>
      <servlet-name>MyCustomGreexInitializer</servlet-name>
      <servlet-class>com.servlet.MyCustomGreexInitializer</servlet-class>
      <load-on-startup>4</load-on-startup>
   </servlet>
   ```

5. 手順 3 で作成したカスタム jar を以下の場所にコピーします。
   - *INSTALL_DIR*/jar/*<application_name>*smcfs
   - *INSTALL_DIR*/repository/eardata/platform/war/yfscommon

     注: この場所にカスタム jar をコピーする前に、これを署名してください。*INSTALL_DIR*/repository/eardata/platform/war/yfscommon/jarlist.txt ファイルを更新する必要もあります。
   - *INSTALL_DIR*/platformrcp/6_0/othertools/com.yantra.ide.othertools.core_1.1.0

Greex 構文のローカライズ

要件に応じて、Greex 構文を適切にローカライズできます。詳しくは、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales : *ローカライズ・ガイド*を参照してください。

Greex ルールに関する情報のロギング

このタスクについて

Greex ルールまたは拡張 XML 条件に関する情報をさまざまなレベルでログに記録できます。情報のロギングのさまざまなレベルについては、GreexLogConstants で説明されています。Apache の log4j フレームワークから単純な System.out.println() の範囲で情報をログに記録できます。

Greex ルールに関する情報をログする手順は、次のとおりです。

手順

1. GreexLogger クラスを作成し、このクラス内に以下のメソッドを実装します。

```
log(GreexLogData data) method and log information as needed.
For example:
public class MyLogger implements GreexLogger
{
    public void log(GreexLogData data)
    {
      System.out.println("Message:: "+data.getMessage());
         System.out.println("Severity:: "+data.getSeverity());
    }
 }
```

2. registerLogger() メソッドを使用して、GreexLogger クラスを GreexContext に登録します。例:

```
public class MyApp
{
 GreexContext  ctx = new GreexContext();
 Ctx.registerLogger(new MyLogger(),GreexLogConstants.GREEX_DEBUG);
}
```

Greex エディター - セットアップ・フェーズ 1

Greex エディターを使用して、拡張 XML ファイル (*.greex ファイルとも呼ばれる) を容易に作成および変更できます。拡張 XML ファイルは、新しい拡張 XML 条件を定義して、入力データに関してその条件を評価するために使用されます。

前提ソフトウェア・コンポーネントのインストール

このセクションでは、Greex エディターを使用して、拡張 XML 条件または Greex ルールを作成するために必要なさまざまなソフトウェア・コンポーネントについて説明します。Greex エディターを使用して拡張 XML 条件を作成する前に、以下のソフトウェア・コンポーネントが既にインストールされていることを確認してください。これらのコンポーネントについて詳しくは、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales: *インストール・ガイド*プラットフォームのインストールを参照してください。

- Eclipse SDK

IBM がサポートしている Eclipse SDK のバージョンをインストールしてください。

- Eclipse 関連のプラグイン

  IBM がサポートしている Eclipse Modeling Framework (EMF) プラグインのバージョンをインストールしてください。

- Java Development Kit (JDK)

  IBM がサポートしている JDK のバージョンをインストールしてください。

  注: JRE バージョン 1.5 より前のブラウザーを使用している場合、Applications ManagerConfigurator を使用して Greex ルールを開こうとすると、例外 java.security.AccessControlException が表示されます。

  このエラーを解決するには、*JRE_HOME*/lib/security/java.policy ファイルで、以下のプロパティーをすべてのドメイン・セクションのデフォルトのアクセス権に追加します。

  ```
  permission java.util.PropertyPermission "java.home", "read";
  ```

  ここで *JRE_HOME* は、JRE のインストール・ディレクトリーです。

  ここで、ブラウザー・インスタンスをすべて閉じ、アプリケーションに再度ログインします。

- Greex プラグイン

  IBM がサポートしているその他のツールのプラグインをインストールしてください。

  このプラグインは、アプリケーションとともに出荷され、*INSTALL_DIR*/platformrcp/6_0/othertools にあります。

Eclipse でのキャッシュ付きビルド情報のクリーニング

このタスクについて

新しいバージョンのその他のツールのプラグインをインストールしているか、または以前のバージョンを更新している場合、Eclipse のキャッシュ付きビルド情報をクリーニングする必要があります。

この情報をクリーニングするには、`-clean` オプションを指定して Eclipse を始動する必要があります。

手順

1. Eclipse のショートカットを右クリックし、ポップアップ・メニューから「**プロパティー**」を選択します。「プロパティー」ウィンドウが表示されます。
2. ターゲットで、コマンド・ライン引数 `-clean` を末尾に入力します。例:

   ```
   "C:\Eclipse 3.2\eclipse\eclipse.exe" -clean.
   ```

   ```
   "C:\Eclipse\eclipse\eclipse.exe" -clean.
   ```

3. Eclipse を始動します。

その他のツールのプラグインのインストール

このタスクについて

その他のツールのプラグインをインストールする手順は、次のとおりです。

手順

1. *INSTALL_DIR*/platformrcp/6_0/othertools ディレクトリー内にあるすべてのフォルダーを *ECLIPSE_HOME*/plugins フォルダーにコピーします。 *ECLIPSE_HOME* は、Eclipse SDK インストール・ディレクトリーです。
2. 新しくインストールしたプラグインの検出を可能にするために Eclipse SDK を再始動します。

Greex エディター - セットアップ・フェーズ 2

このタスクについて

Greex エディターを使用し始める前に、Greex Model ウィザードのコンテナーとして動作する Java プロジェクトを作成する必要があります。

Java プロジェクトを作成する手順は、次のとおりです。

手順

1. Eclipse SDK を始動します。
2. メニュー・バーから、「**ファイル**」 > 「**新規**」 > 「**プロジェクト... (Project...)**」を選択します。「新規プロジェクト (New Project)」ウィンドウが表示されます。
3. ウィザードのリストから、Java カテゴリーで「Java プロジェクト」を選択します。
4. 「**次へ (Next)**」をクリックします。「新規 Java プロジェクト (New Java Project)」ウィンドウが表示されます。
5. プロジェクト名に、新しい Java プロジェクトの名前を入力します。
6. 「**次へ (Next)**」をクリックします。「Java 設定 (Java Settings)」ページが表示されます。
7. 「**終了**」をクリックします。新しい Java プロジェクトが作成されます。

Greex エディター - セットアップ・フェーズ 3

このタスクについて

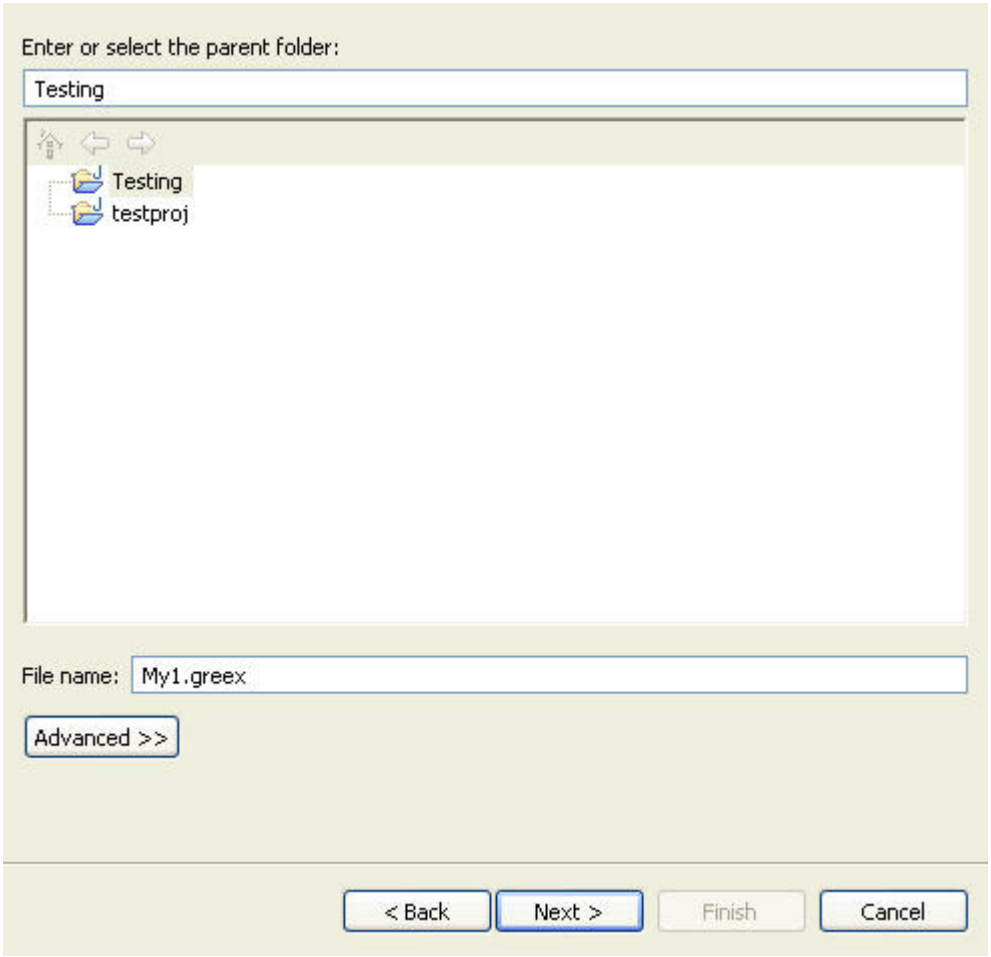
新しい Java プロジェクトを作成した後、作成した新しい Java プロジェクト上で Greex Model ウィザードを実行します。Greex Model ウィザードは、空の *.greex ファイルを作成します。*.greex ファイル内に拡張 XML 条件または Greex ルールを作成できます。

Greex Model ウィザードを実行する手順は、次のとおりです。

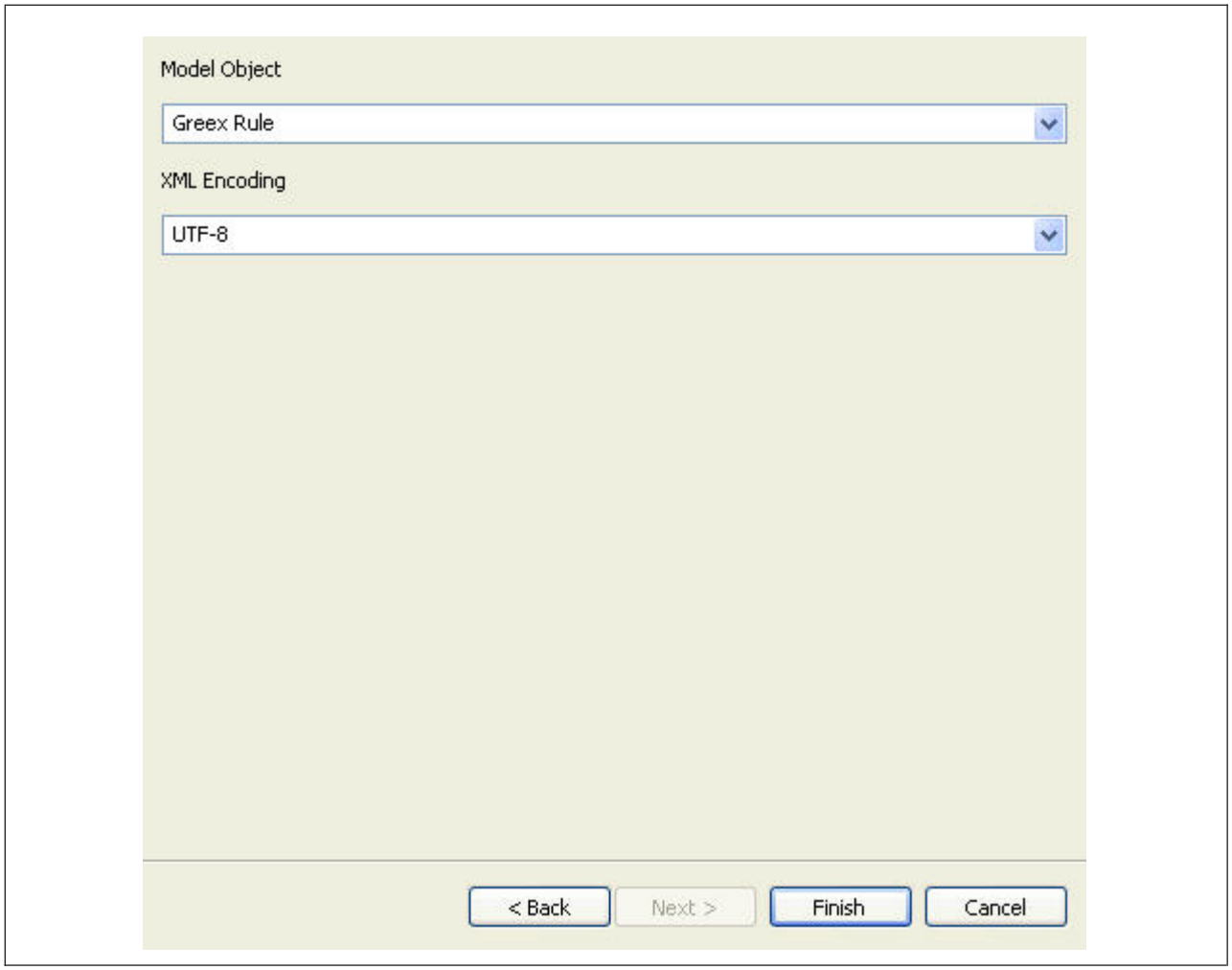
手順

1. Eclipse SDK を始動します。

2. メニュー・バーから、「**ウィンドウ**」 > 「**ビューの表示**」 > 「**ナビゲーター**」を選択します。 Java プロジェクトが「ナビゲーター」ビューに表示されます。
3. 作成した Java プロジェクトを展開します。
4. *.greex ファイルを格納するフォルダーを右クリックし、ポップアップ・メニューから、「**新規**」 > 「**その他 (Other)**」を選択します。 「新規」ウィンドウが表示されます。
5. ウィザードのリストから、「**リッチ・クライアント・プラットフォーム・ウィザード (Rich Client Platform Wizards)**」 > 「**リッチ・クライアント・プラットフォーム Greex Model (Rich Client Platform Greex Model)**」を選択します。
6. 「**次へ (Next)**」をクリックします。 「新規」ウィンドウが表示されます。



7. ファイル名に、*.greex ファイルの名前を入力します。
8. 「**次へ (Next)**」をクリックします。 「Greex Model」ウィンドウが表示されます。



9. 「モデル・オブジェクト (Model Object)」で、「**Greex ルール (Greex Rule)**」を選択します。
10. XML エンコードで、「**UTF-8**」を選択します。
11. 「**終了**」をクリックします。 *.greex ファイルが指定したフォルダーに作成されます。

新しい拡張 XML 条件の作成

*.greex ファイルを作成した後、拡張 XML 条件または Greex ルールを定義できます。以下の 2 つのタイプの拡張 XML 条件を作成できます。

- 通常拡張 XML 条件
- 決定表ベースの拡張 XML 条件

通常拡張 XML 条件

*.greex ファイルを作成した後、通常拡張 XML 条件または Greex ルールを定義できます。

注: あらゆるタイプの拡張 XML 条件または Greex ルールにコメントを追加していないことを確認してください。

通常拡張 XML 条件は、拡張 XML 条件に対して複数の条件を定義する必要があるが、それぞれの条件が異なる属性に関連付けられているシナリオで便利です。これらの複数の条件は、ネスト化された IF および ELSE の構造を使用して定義されます。例えば、以下のような条件の通常拡張 XML 条件を作成する必要があるとします。

```
If Ordertype="WEB" and OrderQty="100", then TaxExemptFlag="N".
Else
   If OriginalTotalAmount="1000" and OrderLine>"5", then
      discount="10" TaxExemptFlag="Y".
   Else TaxExemptFlag="N".
```

この場合、複数の条件があり、それぞれの条件が異なる属性に関連付けられています。1 つ目の条件は、Ordertype、OrderQty、および Discount に関連付けられています。一方 2 つ目の条件は、OriginalTotalAmount、OrderLine、および TaxExemptFlag に関連付けられています。

これらの条件は、ネスト化された IF および ELSE の構造を使用して *.greex ファイルで定義されます。拡張 XML 条件の *.greex ファイルのサンプルを以下に示します。

```
<GreexRule desc="Determine Discount and Tax Exemption based on
                  various attributes"
         id="getDiscountTaxExempt"
         name="Get Discount and Tax Exempt"
         returnType="Xml"
         type="">
<If>
   <Condition name="isDiscount20?">
     <Group op="and">
       <Expression>fn:equals(@orderType, &quot;WEB&quot;)</Expression>
       <Expression>fn:equals(@orderQty, &quot;100&quot;)</Expression>
     </Group>
   </Condition>
   <Return>
     <Value output="&lt;Order TaxExemptFlag=&quot;N&quot;/>"/>
   </Return>
</If>
<Else>
  <If>
    <Condition name="TaxExempted?">
     <Group op="and">
       <Expression>fn:equals(@OriginalTotalAmount, &quot;1000&quot;)</Expression>
       <Expression>fn:intGreater(@orderLine,&quot;5&quot;)</Expression>
     </Group>
    </Condition>
    <Return>
       <Value output="&lt;Order discount=&quot;10&quot;
                      TaxExemptFlag=&quot;Y&quot;/>"/>
    </Return>
   </If>
<Else>
   <Return>
     <Value output="&lt;Order discount=&quot;20&quot;
                    TaxExemptFlag=&quot;N&quot;/>"/>
   </Return>
</Else>
</Else>
</GreexRule>
```

決定表ベースの拡張 XML 条件

*.greex ファイルを作成した後、通常拡張 XML 条件または Greex ルールを定義できます。

注: あらゆるタイプの拡張 XML 条件または Greex ルールにコメントを追加していないことを確認してください。

決定表ベースの拡張 XML 条件は、拡張 XML 条件に対して複数のネスト化された条件を定義するが、それぞれの条件が同じ属性に関連付けられているシナリオで便利です。このような場合、1 つの条件のみを書き込み、切り替え文のように動作するパラメーターのテーブルを用意できます。決定表ベースの拡張 XML 条件では、単一の IF 構造で複数のネスト化された条件を定義できます。したがって、決定表ベースの拡張 XML 条件では、ELSE 構造はありません。IF 構造には、通常拡張 XML 条件で使用されるような 1 つの定数値の代わりに定数値の配列が含まれます。IF 構造には、切り替え文のように動作するパラメーターのテーブルがあります。

例えば、以下のような条件の決定表ベースの拡張 XML 条件を作成する必要があるとします。

```
If Ordertype="WEB" and OrderLineQty="200", then Discount="5".
Else
   If Ordertype="STORE" and OrderLineQty="500", then  Discount="7"
Else
  If Ordertype="CALL" and OrderLineQty="250", then Discount="3".
  Else
   default="0"
```

以下の決定表は、前のシナリオを説明しています。

オーダー・タイプ	オーダー明細の数量	割引
WEB	200	5
STORE	500	7
CALLCENTER	250	3
デフォルト 0		

この場合、複数の条件があり、それぞれの条件が同じ属性に関連付けられています。すべての条件が Ordertype、OrderLineQty、Discount の各属性に関連付けられています。したがって、この拡張 XML 条件には、IF 構造が 1 つしかなく、定数値の配列が含まれています。

注: 決定表ベースの Greex ルールの場合、デフォルトの戻り値を指定する必要があります。この値は、満たされる IF 条件がない場合に返されます。

これらの条件は、IF 構造を使用して *.greex ファイルで定義されます。決定表ベースの拡張 XML 条件の *.greex ファイルのサンプルを以下に示します。

```
<GreexRule desc="Determine Discount based on some attributes"
           id="getDiscount"
           name="Get Discount"
           returnType="String"
           type="DecisionTable">>
 <If>
```

```
  <Condition name="isDiscount5?">
    <Group op="and">
      <Expression>fn:equals(@orderType,
                &amp;quot;WEB|STORE|CALLCENTER&amp;quot;)</Expression>
      <Expression>fn:equals(@orderLineQty, &amp;quot;200|500|250&amp;quot;)</Expression>
    </Group>
  </Condition>
  <Return>
     <Value default="0" output="5|7|3"/>
   </Return>
 </If>
</GreexRule>
```

通常拡張 XML 条件の作成

このタスクについて

通常拡張 XML 条件を作成する手順は、次のとおりです。

手順

1. 作成した Java プロジェクトを展開します。
2. プロジェクト・エクスプローラー階層で、*.greex ファイルを選択します。ポップアップ・メニューから「**次のツールで開く (Open With)**」 > 「**Greex Model エディター (Greex Model Editor)**」を右クリックして選択します。
3. 「Greex エディター」に表示されるツリー構造を展開します。ツリーからノードをクリックします。 ノードのすべての子リーフがリストされます。「文書ルート (Document Root)」要素には、「Greex ルール・ルート要素 (Greex Rule root element)」が含まれています。
4. 「プロパティー」ビューで「Greex ルール・ルート要素 (Greex Rule root element)」を選択します。さまざまな属性の値を入力します。 「Greex ルール・ルート要素 (Greex Rule root element)」のさまざまな属性の説明については、以下のテーブルを参照してください。
5. 「プロパティー」ビューで、選択した要素のさまざまなプロパティーを表示できます。「プロパティー」ビューを開く手順は、次のとおりです。
   a. メニュー・バーから、「**ウィンドウ**」 > 「**ビューの表示**」 > 「**その他... (Other...)**」を選択します。 ビューのリストの「基本」の下から、「**プロパティー**」を選択します。

属性	説明
Desc	Greex ルールの説明を入力します。
ID	Greex ルールの固有 ID を入力します。
Name	Greex ルールの名前を入力します。
返品タイプ	Greex ルールの戻りタイプを入力します。通常 Greex ルールの場合、有効な値は、「Xml」、「String」、「Boolean」、および「PureXML」です。通常 Greex ルールは、XML 文書、ストリング、またはブール値を返すことができます。
型	デフォルトでは、Greex ルールは通常 Greex ルールです。

6. 「Greex ルール・ルート要素 (Greex Rule root element)」で、必要に応じて、新しい IF ELSE 構造要素を作成します。「**Greex ルール・ルート要素 (Greex**

Rule root element)」を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**If/Else**」を選択します。 任意のレベルのネスト化された IF および ELSE の構造を作成できます。

7. 「**If/Else**」要素を選択します。「プロパティー」ビューで「If」要素の名前を「名前」プロパティーに入力します。
8. 「If/Else」要素で、新しい条件の子要素を作成します。「**If/Else**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**条件**」を選択します。
9. 「条件」要素を選択します。「プロパティー」ビューで「条件」要素の名前を「名前」プロパティーに入力します。
10. それぞれの条件が値を返す必要があるため、「If/Else」要素で新しい「返品」要素を作成し、関連条件に対する適切な戻り値を指定します。「**If/Else**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」>「**返品**」を選択します。
11. 「**返品**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」>「**値**」を選択します。ポップアップ・ウィンドウが表示する値を入力します。
12. IF 条件が満たされたときに返す値を入力します。
13. 「**返品**」要素を選択します。「プロパティー」ビューの「デフォルト・プロパティー (Default property)」に (必要に応じて) IF 条件が満たされない場合に返すデフォルト値を指定します。「出力プロパティー」で、IF 条件が満たされたときに返す値を指定します。
14. 「条件」要素で、新しい「式 (Expression)」要素を作成し、満たすべき条件を評価する式を指定します。「**条件**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**式 (Expression)**」を選択します。「式の編集 (Edit Expression)」ポップアップ・ウィンドウが表示されます。
15. 「式 (Expression)」に、評価する式を入力します。式で関数呼び出しを行うには、関数名の先頭に「`fn:`」を付けます。関数を他の関数に渡すこともできます。

    注: 「**Ctrl+ スペース (Ctrl+Space)**」を押して、ドロップダウン・リストから式を選択します。
16. (オプション) 式のセットを一緒に評価する場合、その式のセットをグループ化する必要があります。「条件」要素で、新しい「グループ」要素を作成し、式のセットをグループ化します。「**条件**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**グループ**」を選択します。

    ここで、このグループに複数の式を追加できます。「**グループ**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**式 (Expression)**」を選択します。

    新しい「グループ」要素を既存の「グループ」要素に追加することも可能です。「**グループ**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**グループ**」を選択します。

    注: 任意のレベルのネスト化された「グループ」および「式 (Expression)」の要素を作成できます。

17. (オプション)「グループ」要素を選択します。「プロパティー」ビューで、「Op」プロパティーの式のセットに対して実行する命令を指定します。有効な値は、「or」および「and」です。

    Op プロパティーを「or」として指定すると、グループで指定した式のいずれかの評価が「true」の場合に条件が満たされます。

    Op プロパティーを「and」として指定すると、グループで指定した式のすべての評価が「true」の場合のみ条件が満たされます。

18. 「**保存**」をクリックします。

決定表ベースの拡張 XML 条件の作成

このタスクについて

新しい決定表ベースの拡張 XML 条件を作成する手順は、次のとおりです。

手順

1. 作成した Java プロジェクトを展開します。
2. プロジェクト・エクスプローラー階層で、*.greex ファイルを選択します。「**次のツールで開く (Open With)**」 > 「**Greex Model エディター (Greex Model Editor)**」を右クリックします。
3. 「Greex エディター」パネルに表示されるツリー構造を展開します。ツリーからノードをクリックします。 ノードのすべての子リーフがリストされます。「文書ルート (Document Root)」要素には、「ルート要素 Greex ルール (root element Greex Rule)」が含まれています。
4. 「ルート要素 Greex ルール (root element Greex Rule)」を選択し、「プロパティー」ビューでさまざまな属性の値を入力します。

    以下のテーブルでは、「ルート要素 Greex ルール (root element Greex Rule)」のさまざまな属性について説明しています。

    「プロパティー」ビューで、選択した要素のさまざまなプロパティーを表示できます。「プロパティー」ビューを開くには、メニュー・バーから、「**ウィンドウ**」 > 「**ビューの表示**」 > 「**その他... (Other...)**」を選択します。ビューのリストの「基本」の下から、「**プロパティー**」を選択します。

属性	説明
Desc	Greex ルールの説明を入力します。
ID	Greex ルールの固有 ID を入力します。
Name	Greex ルールの名前を入力します。
返品タイプ	Greex ルールの戻りタイプを入力します。決定表ベースの Greex ルールの場合、有効な値は「ストリング」のみです。決定表ベースの Greex ルールは、ストリングのみを返すことができます。
型	Greex ルールのタイプを入力します。デフォルトでは、通常 Greex ルールです。これを決定表ベースの Greex ルールにするには、ドロップダウン・リストから「決定表」を選択します。

5. 「ルート要素 Greex ルール (root element Greex Rule)」で、要件に従って新しい IF 構造要素を作成します。「**ルート要素 Greex ルール (root element Greex Rule)**」を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**If**」を選択します。

   注: 決定表ベースの Greex ルールは ELSE 構造を含むことはできません。単一の IF 構造を含むことができるだけです。
6. 「**If**」要素を選択し、「プロパティー」ビューで「If」要素の名前を「名前」プロパティーに入力します。
7. 「If」要素で、新しい条件の子要素を作成します。「**If**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**条件**」を選択します。
8. 「**条件**」要素を選択し、「プロパティー」ビューで「条件」要素の名前を「名前」プロパティーに入力します。
9. それぞれの条件が値を返す必要があるため、「If」要素に新しい「返品」要素を作成し、「値」要素の関連条件に対する適切な戻り値を指定します。「**If**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**返品**」を選択します。
10. 「**返品**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**値**」を選択します。ポップアップ・ウィンドウが表示する値を入力します。
11. IF 条件が満たされたときに返す値を入力します。
12. 「**返品**」要素を選択します。「プロパティー」ビューの「デフォルト・プロパティー (Default property)」に IF 条件のいずれも満たされない場合に返すデフォルト値を指定し、「出力プロパティー」に IF 条件が満たされたときに返す値を指定します。

    注: 決定表ベースの Greex ルールの場合、デフォルトの戻り値を指定する必要があります。

    決定表ベースの Greex ルールの場合、「値」要素に定数値の配列を定義できます。複数の値は、「|」演算子を使用して分離します。例えば、`5|7|3` のようにします。
13. 「条件」要素で、新しい「式 (Expression)」要素を作成し、満たされるべき条件を評価する式を指定します。「**条件**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**式 (Expression)**」を選択します。「式の編集 (Edit Expression)」ポップアップ・ウィンドウが表示されます。
14. 「式 (Expression)」に、評価する式を入力します。

    式で関数呼び出しを行うには、関数名の先頭に「fn:」を付けます。関数を他の関数に渡すこともできます。

    注: 「**Ctrl+ スペース (Ctrl+Space)**」を押して、ドロップダウン・リストから式を選択できます。

    決定表ベースの Greex ルールの場合、「式」要素の特定の XML 属性に定数値の配列を定義できます。複数の値は、「|」演算子を使用して分離します。例えば、`fn:equals(@orderType, "WEB|STORE|CALLCENTER")` のようにします。

15. (オプション) 式のセットを一緒に評価する場合、式のセットをグループ化する必要があります。「条件」要素で、新しい「グループ」要素を作成し、式のセットをグループ化します。「**条件**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**グループ**」を選択します。

    ここで、このグループに複数の式を追加できます。「**グループ**」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**式 (Expression)**」を選択します。

    新しい「グループ」要素を「グループ」要素に追加することも可能です。「グループ」要素を右クリックし、ポップアップ・メニューから「**新しい子 (New Child)**」 > 「**グループ**」を選択します。

    注: 「グループ」要素および「式 (Expression)」要素のあらゆるレベルのネスト化が許可されます。
16. (オプション)「**グループ**」要素を選択し、「プロパティー」ビューで、「op」プロパティーの式のセットに対して実行する命令を指定します。有効な値は、「or」および「and」です。

    Op プロパティーを「or」として指定すると、グループで指定した式のいずれかの評価が true の場合に条件が満たされます。

    Op プロパティーを「and」として指定すると、グループで指定した式のすべての評価が true の場合のみ条件が満たされます。
17. 「**保存**」ボタンをクリックして、変更を保存します。

拡張 XML 条件の検証

このタスクについて

拡張 XML 条件を作成した後、その構文および構造を検証する必要があります。例えば、拡張 XML 条件が決定表タイプの場合、1 つの IF 構造のみを含む必要があります。ELSE 構造を含むことはできません。したがって、拡張 XML 条件をデータベースにロードする前に、これを検証する必要があります。

拡張 XML 条件を検証する手順は、次のとおりです。

手順

1. 作成した Java プロジェクトを展開します。
2. プロジェクト・エクスプローラー階層で、*.greex ファイルを選択します。ポップアップ・メニューから「**次のツールで開く (Open With)**」 > 「**Greex Model エディター (Greex Model Editor)**」を右クリックして選択します。
3. 「Greex エディター」パネルに表示される Greex ルールを展開します。ツリーからノードをクリックします。 ノードのすべての子リーフがリストされます。「文書ルート (Document Root)」要素には、「Greex ルール・ルート要素 (Greex Rule root element)」が含まれています。
4. 「Greex ルール・ルート要素 (Greex Rule root element)」を選択し、右クリックします。ポップアップ・メニューから「**Greex ルールを検証する (Validate Greex Rule)**」を選択します。 作成した Greex ルールが正しい構文および構造

を使用している場合、メッセージ「Greex ルールの検証に成功しました (Greex rule validation succeeded)」が表示されます。それ以外の場合、システムに適切なエラー・メッセージが表示されます。

拡張 XML 条件のデータベースへのロード

リッチ・クライアント・プラットフォーム Greex エディターを使用して拡張 XML 条件または Greex ルールを作成した後、*.greex ファイルをデータベースにロードします。このファイルには、データベースにロードする、新しく作成した拡張 XML 条件が含まれています。Applications ManagerConfigurator を使用して、(必要に応じて) 拡張 XML 条件の適切なパラメーターを変更できます。

注: 拡張 XML 条件または Greex ルールをデータベースにロードする前に、拡張 XML 条件が有効な構造または構文かどうか検証する必要があります。詳しくは、を参照してください。.

拡張 XML 条件をインポートまたはロードする前に

手順

1. *ECLIPSE_HOME*/plugins/com.yantra.ide.othertools.greex.editor_*VERSION_NUMBER*/lib/greexide.properties ファイルを変更し、使用しているデータベースに基づいて、JDBC 接続プロパティーを指定します。 *ECLIPSE_HOME* は、Eclipse SDK をインストールしたディレクトリーです。*VERSION_NUMBER* は、Greex エディター・プラグインの現行バージョン番号です。
2. データベースに応じて、必要なデータベース・ドライバー JAR ファイルを *ECLIPSE_HOME*/plugins/com.yantra.ide.othertools.greex.editor_*VERSION_NUMBER*/lib ディレクトリーにコピーします。 例えば、Oracle データベースを使用している場合、ojdbc14.jar ファイルをコピーします。
3. *ECLIPSE_HOME*/plugins/com.yantra.ide.othertools.greex.editor_*VERSION_NUMBER*/META-INF/MANIFEST.MF ファイルを編集して、必要なデータベース・ドライバー JAR ファイルの相対パス (*ECLIPSE_HOME*/plugins/com.yantra.ide.othertools.greex.editor_*VERSION_NUMBER* ディレクトリーへの相対パス) を Bundle-ClassPath プロパティーに追加します。

   例えば、Oracle データベースを使用している場合、次のように `lib/ojdbc14.jar` を Bundle-ClassPath プロパティーに追加します。

   ```
   Bundle-ClassPath: greexeditor.jar, lib/ojdbc14.jar
   ```
4. Eclipse SDK を既に実行中の場合、次のように Eclipse をクリーン・モードで再始動します。

   ```
   eclipse.exe -clean
   ```

拡張 XML 条件をロードする手順

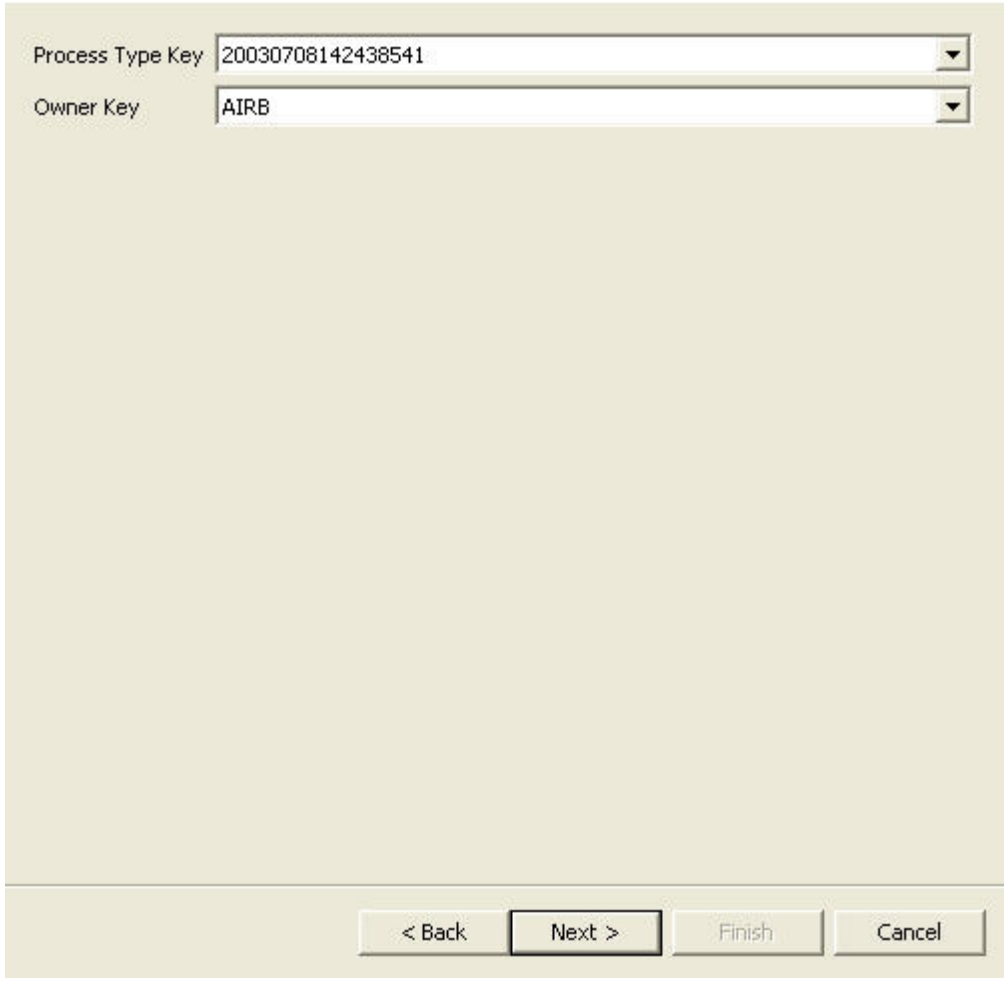
このタスクについて

拡張 XML 条件をデータベースにロードする手順は、次のとおりです。

手順

1. Eclipse SDK を始動します。

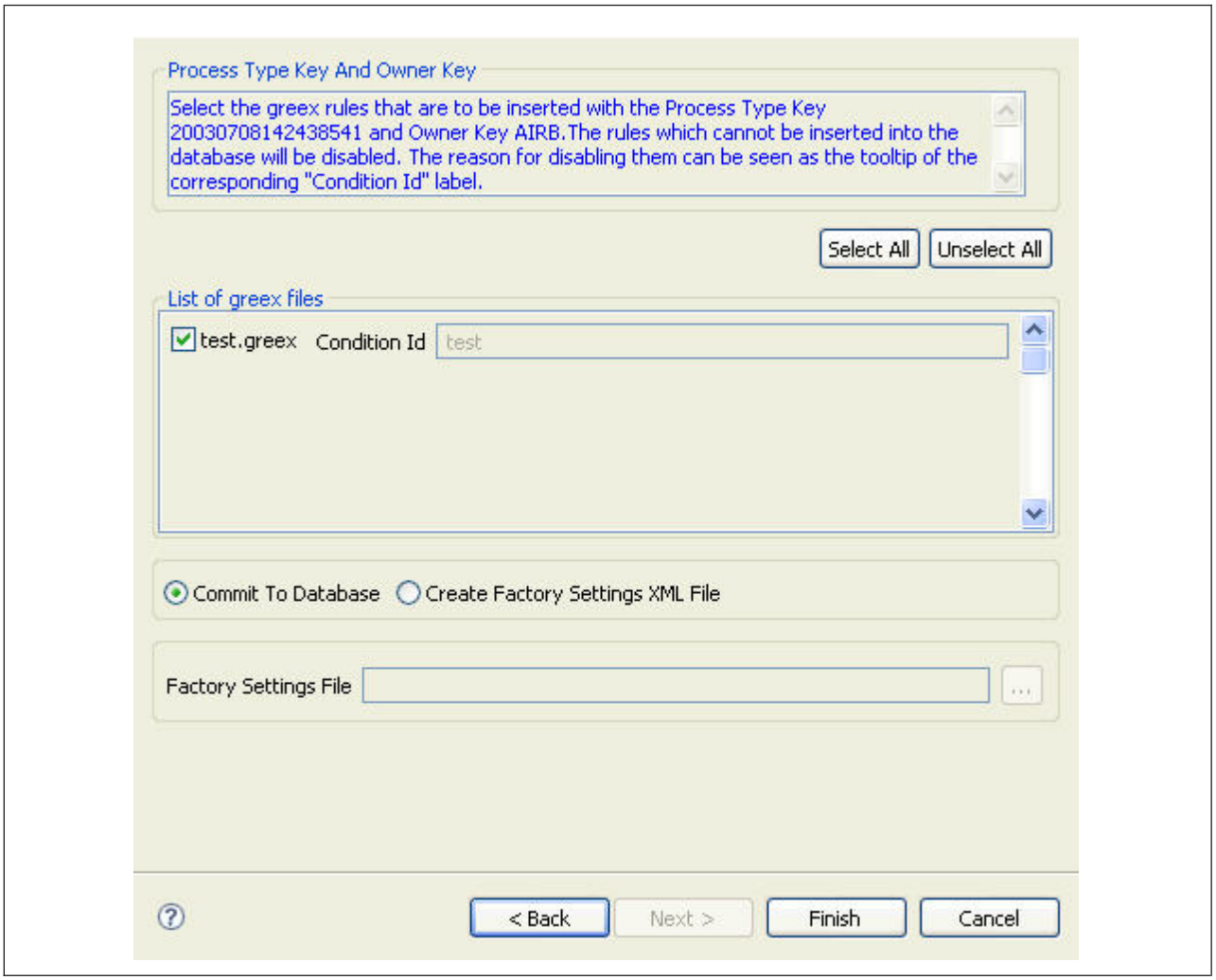
2. メニュー・バーから、**「ウィンドウ」 > 「ビューの表示」 > 「ナビゲーター」**を選択します。 Java プロジェクトが「ナビゲーター」ビューに表示されます。
3. 作成した Java プロジェクトを展開します。
4. データベースにロードする拡張 XML 条件が含まれているすべての *.greex ファイルを含むフォルダーを右クリックします。ポップアップ・メニューから**「Greex」 > 「Greex ルールをデータベースにエクスポートする (Export Greex Rule to Database)」**を選択します。 「Greex ルールをデータベースにエクスポートする (Export Greex Rule to Database)」ウィンドウが表示されます。



フィールド	説明
「プロセス・タイプ・キー (Process Type Key)」	ドロップダウン・リストから適切なプロセス・タイプのキーを選択します。
「所有者キー」	ドロップダウン・リストから適切な所有者のキーを選択します。

5. **「次へ (Next)」**をクリックします。

「Greex ルールの選択 (Select Greex Rules)」ウィンドウが表示されます。



フィールド	説明
「プロセス・タイプ・キー (Process Type Key) および所有者キー」	選択したプロセス・タイプ・キーおよび所有者キーが表示されます。
「すべて選択」	「Greex ファイル (Greex Files)」パネルにリストされているすべての *.greex ファイルを選択する場合、このボタンをクリックします。
「すべて選択解除」	「Greex ファイル (Greex Files)」パネルにリストされているすべての *.greex ファイルを選択解除する場合、このボタンをクリックします。
「Greex ファイルのリスト (List of Greex files)」	データベースにロードする *.greex ファイルを選択します。
「接続 ID」	選択した各 *.greex ファイルで定義された Greex ルールの固有 ID を表示します。

フィールド	説明
「データベースにコミット (Commit To Database)」	選択した Greex ルールをデータベースに対してコミットする場合、このオプションを選択します。
「工場出荷時設定 XML ファイルを作成する (Create Factory Settings XML File)」	選択した Greex ルールに対して工場出荷時設定 XML ファイルを作成する場合、このオプションを選択します。
「工場出荷時設定ファイル (Factory Settings File)」	YFS_CONDITION 工場出荷時設定を格納する新しい工場出荷時設定 XML ファイルのパスと名前を入力します。  をクリックします。「XML ファイルを YFS_CONDITION 工場出荷時設定に格納することを選択する (Select XML File to Store the YFS_CONDITION Factory Settings)」ポップアップ・ウィンドウが表示されます。YFS_CONDITION 工場出荷時設定を格納する工場出荷時設定 XML ファイルを選択します。

6. 「**終了**」をクリックします。

データベースからの拡張 XML 条件のインポート

既存の拡張 XML 条件または Greex ルールをデータベースからローカル・マシンにインポートできます。これは、拡張 XML 条件の構造を変更 (新しい式または IF および ELSE の構造を既存の拡張 XML 条件に追加するなど) する場合に便利です。

拡張 XML 条件をインポートまたはロードする前に

手順

1. *ECLIPSE_HOME*/plugins/com.yantra.ide.othertools.greex.editor_*VERSION_NUMBER*/lib/greexide.properties ファイルを変更し、使用しているデータベースに基づいて、JDBC 接続プロパティーを指定します。 *ECLIPSE_HOME* は、Eclipse SDK をインストールしたディレクトリーです。*VERSION_NUMBER* は、Greex エディター・プラグインの現行バージョン番号です。
2. データベースに応じて、必要なデータベース・ドライバー JAR ファイルを *ECLIPSE_HOME*/plugins/com.yantra.ide.othertools.greex.editor_*VERSION_NUMBER*/lib ディレクトリーにコピーします。 例えば、Oracle データベースを使用している場合、ojdbc14.jar ファイルをコピーします。
3. *ECLIPSE_HOME*/plugins/com.yantra.ide.othertools.greex.editor_*VERSION_NUMBER*/META-INF/MANIFEST.MF ファイルを編集して、必要なデータベース・ドライバー JAR ファイルの相対パス (*ECLIPSE_HOME*/plugins/com.yantra.ide.othertools.greex.editor_*VERSION_NUMBER* ディレクトリーへの相対パス) を Bundle-ClassPath プロパティーに追加します。

   例えば、Oracle データベースを使用している場合、次のように `lib/ojdbc14.jar` を Bundle-ClassPath プロパティーに追加します。

   ```
   Bundle-ClassPath: greexeditor.jar, lib/ojdbc14.jar
   ```

4. Eclipse SDK を既に実行中の場合、次のように Eclipse をクリーン・モードで再始動します。

   ```
   eclipse.exe -clean
   ```

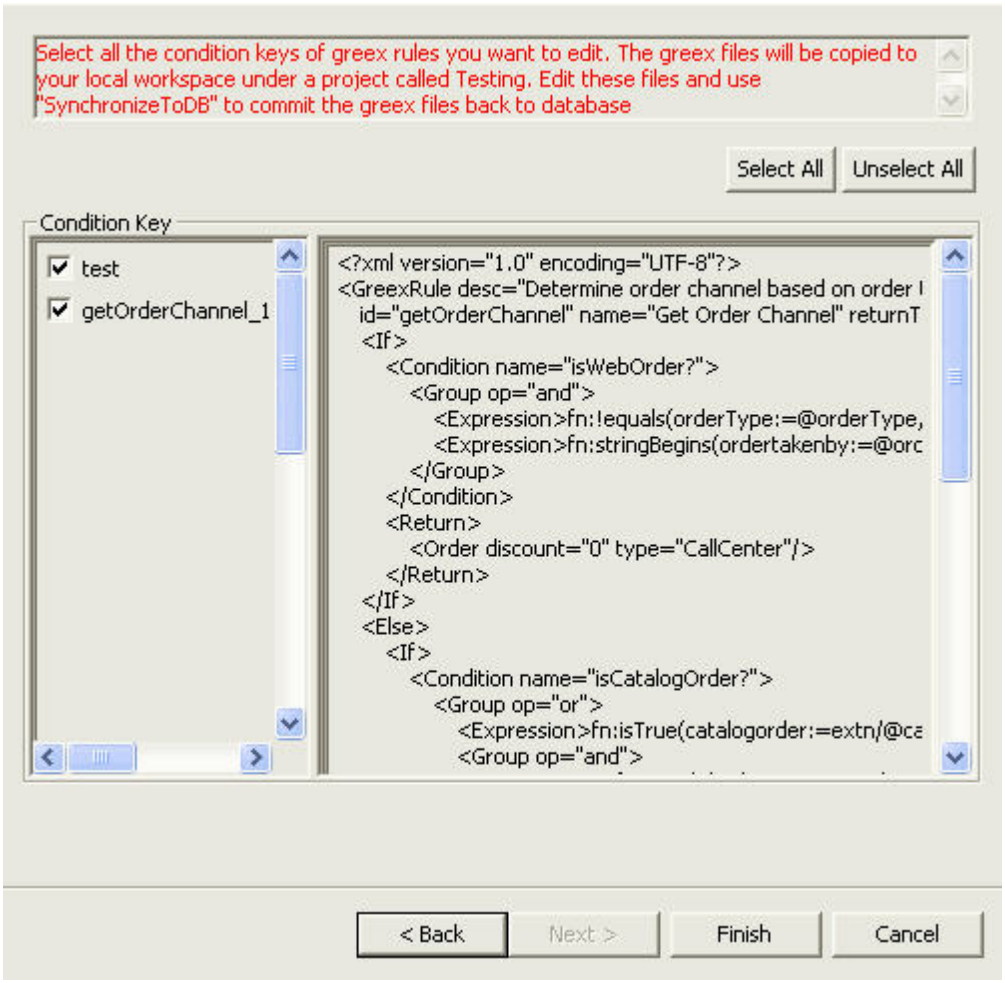
拡張 XML 条件をインポートする手順

このタスクについて

拡張 XML 条件をデータベースからインポートする手順は、次のとおりです。

手順

1. Eclipse SDK を始動します。
2. メニュー・バーから、「**ウィンドウ**」 > 「**ビューの表示**」 > 「**ナビゲーター**」を選択します。Java プロジェクトが「ナビゲーター」ビューに表示されます。
3. データベースから *.greex ファイルにインポートする Java プロジェクトを右クリックします。
4. ポップアップ・メニューから「**Greex**」>「**Greex ルールをデータベースからインポートする (Import Greex Rule from Database)**」を選択します。 「Greex ルールをデータベースからインポートする (Import Greex Rule from Database)」ウィンドウが表示されます。各 *.greex ファイルには、拡張 XML 条件または Greex ルールが含まれています。



フィールド	説明
「すべて選択」	「条件キー (Condition Key)」パネルにリストされているすべての *.greex ファイルを選択する場合、このボタンをクリックします。
「すべて選択解除」	「条件キー (Condition Key)」パネルにリストされているすべての *.greex ファイルを選択解除する場合、このボタンをクリックします。
「条件キー」	データベースにロードされる各 *.greex ファイルに対して指定された条件 ID のリストを表示します。特定の条件 ID の上にマウスを置くと、右パネルに、選択した拡張 XML 条件または Greex ルールの内容が表示されます。 例えば、Greex ルールまたは拡張 XML 条件の ID、名前、説明などです。この拡張 XML 条件に対して定義されている IF および ELSE の構造もすべて表示されます。ローカル・マシンにコピーしない *.greex ファイルを適切に選択または選択解除します。

5. 「**終了**」をクリックします。 選択した *.greex ファイルは、Java プロジェクトにインポートされます。

6. Greex エディターを使用して *.greex ファイルを開き、必要に応じて拡張 XML 条件または Greex ルールを変更します。
7. 拡張 XML 条件をロードする手順の説明に従って、変更した *.greex ファイルをデータベースに再ロードします。

拡張 XML 条件の変更

Applications ManagerConfigurator を使用して、拡張 XML 条件の変更可能パラメーターのみを変更できます。拡張 XML 条件の構造は変更できません。拡張 XML 条件の変更について詳しくは、Sterling Business CenterSterling Selling and Fulfillment FoundationSterling Field Sales: 構成ガイドを参照してください。

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