ORACLE®

ATG WEB COMMERCE

バージョン 11.0

ビジネス・コマース・リファレンス アプリケーション・ガイド

Oracle ATG
One Main Street
Cambridge, MA 02142
USA

Oracle Commerce ビジネス・コマース・リファレンス・アプリケーション・ガイド

ドキュメントのバージョン

Doc11 BCOMMREFv1 4/15/2011

Copyright

目次

1 はじめに

Oracle Commerce ビジネス・コマース・リファレンス・アプリケーションは、B2B サイトで役立つ Oracle Commerce Core Commerce の各種機能を紹介します。

Oracle Commerce ビジネス・コマース・リファレンス・アプリケーションには、Motorprise という架空の会社の、パーソナライズされたビジネス・コマース・サイトが用意されています。個々のユーザーがパーソナライズされた、革新的な、コマース中心の Web サイトを迅速かつ円滑に作成するために役立つ基礎およびガイドとなるように、Core Commerce の豊富な機能が使用されています。

Motorprise は事前に構成された Java コンポーネントとページ・テンプレートで構成されており、これにより、オンライン・コマース・サイトに特有の、様々なマーチャンダイジングおよびオーダー処理のアクティビティを迅速に実装するために必要なコア機能が提供されます。Core Commerce で使用可能な基本機能に加え、機能拡張を加えることによって、請求書/購買オーダー支払方法、承認、コスト・センター、繰返しおよび定期の購買依頼など、特定の機能として実装することができます。

このマニュアルで用意されているコード・サンプルは機能を理解するためにも有用で、さらにコード・サンプルを要件にあわせて応用することによって実装プロジェクトに活用することもできます。このマニュアルでは、Core Commerce 機能への追加や拡張についても説明します。サイトのコマース領域はモジュール形式で構築されており、相互依存性が最小限のため、サイトを分割して個々に再実装できます。

ユーザーは自分のビジネス目標を評価し、その戦略に適した製品機能を使用できます。このマニュアルに記載されているプロセスと機能は、各自の Web アプリケーションの特定のニーズにあわせてカスタマイズできます。ページ開発者、プログラマおよびビジネス・ユーザーは、このマニュアルを参照して、使用できる機能を確認し、各自のサイトにあわせてこれらの機能をカスタマイズする最も効果的な方法を決定する必要があります。

Motorprise を操作する前に、『ATG Web Commerce Programming Guide』および『ATG Web Commerce ストア設定コマース・ガイド』に記載されている概念を理解している必要があります。

この章の以降では、Motorprise サイトについて理解し、開始するために必要な基本情報について説明します。

概要

Motorprise サイトをインストールする方法は、『ATG Web Commerce Programming Guide』に記載されています。Oracle Commerce ビジネス・コマース、Motorprise および ATG Control Center（ACC）を含むアプリケーションのアセンブルについては、『ATG Web Commerce Platform Programming Guide』を参照してください。

E メールの設定

顧客が Motorprise から E メールを受信できるように Motorprise サイトを構成するには、E メール・ホストを有効なサーバーに設定し、いくつかのシナリオを有効にします。

最初に、E メール・サーバーを指定します。

1. 「ページおよびコンポーネント」→「コンポーネント」で ATG Control Center のパス・メニューにより、/atg/dynamo/service/SMTPEmail に進みます。
2. emailHandlerHostName プロパティを有効なメール・サーバーに設定します。

次に、オーダーが履行および配信されるときに E メールを送信するシナリオを有効にします。

1. 「シナリオ」→「シナリオ」メニューに移動します。
2. 次のシナリオを選択し、有効にします。

   DCS/Fulfillment

   DCS/ReceiveOrder

   B2B/25OverAverage

   B2B/ApprovalNotification

   B2B/NewUserRegistered

   B2B/OneMonth10Percent

注意: メール・サーバーが送信者の E メール・アドレスが有効かどうかをチェックする場合は、<ATG11dir>/MotorpriseJSP/j2ee-apps/Motorprise/web-app/en/email および <ATG11dir>/MotorpriseJSP/j2ee-apps/Motorprise/web-app/de/email にある、すべての E メール・テンプレート内の送信者メール・アドレスを変更する必要があります。

Motorprise からのメッセージを受信して読む場合は、ユーザー・プロファイル内の E メール・アドレスを変更して、<user>@example.com ではなく実際の E メール・アドレスを使用する必要があります。

サイトの表示

Oracle Commerce Core Platform、Core Commerce および Motorprise を含むアプリケーションをアセンブルして配置すると、次の URL を使用してアクセスできます。

```
http://hostname:port/Motorprise
```

使用するポートは、アプリケーション・サーバーの構成に依存します。たとえば、JBoss のデフォルト URL は次のとおりです。

```
http://hostname:8080/Motorprise
```

アプリケーション・アセンブリと Oracle Commerce Platform モジュールの詳細は、『ATG Web Commerce Platform Programming Guide』を参照してください。デフォルト・ポートの詳細は、『ATG Web Commerce Installation and Configuration Guide』を参照してください。

2 ビジネス・ユーザーの概要

Oracle Commerce ビジネス・コマース・リファレンス・アプリケーションは、B2B のコマース Web サイトのデモ・サンプルです。

Motorprise Inc.は自動車部品の大手販売業者です。Motorprise サイトは、次のものを備えています。

- 組織階層、個人、ロールおよび継承を作成できる機能
- Oracle Commerce Core Commerce 承認ワークフローを使用する承認プロセス
- 契約に基づく複数の顧客固有のカタログおよび価格表
- 数量価格決定やオーダー制限など、複雑な購買要件のサポート
- 支払方法としての購買オーダー、購買依頼およびクレジット・カード
- 明細品目ごとに複数の支払方法、出荷先所在地およびコスト・センターを使用する機能
- 保存されたオーダーと定期オーダーのサポート
- 堅牢な製品検索および比較
- Motorprise 顧客がそのユーザーおよび承認者を Web ブラウザを通じて管理できる機能
- SOAP と XML を介した統合ポイント
- シナリオの例

これらの各機能の実装方法の技術的な詳細は、後の章を参照してください。

カタログの設定

Motorprise サイトは、ビジネス・パートナ（Motorprise から大量の自動車部品を購入する顧客）を対象としています。

次の目的でカタログが設定されています。

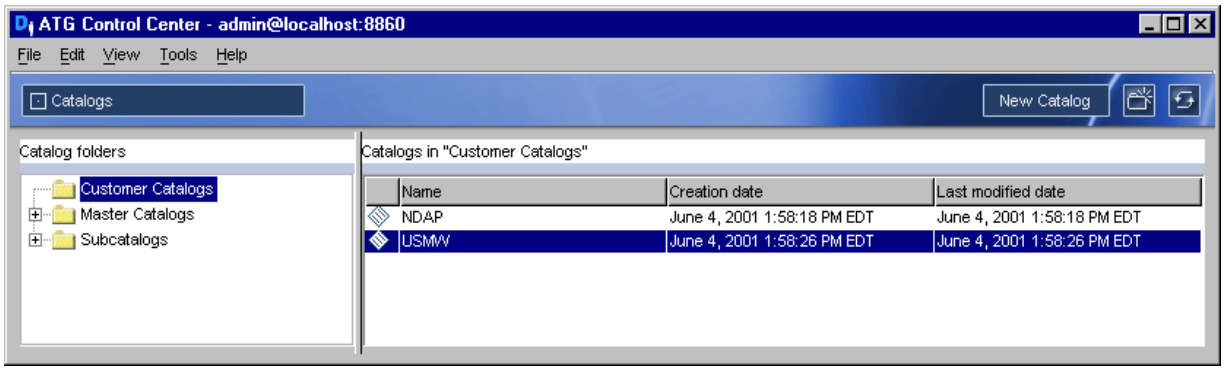
- 特定の製品グループを特定の顧客に簡単に表示します
- 顧客間でカテゴリを共有できるため、顧客ごとに個別のカタログを作成する必要がありません
- 同じカテゴリを使用して、マスター・カタログを作成します
- Motorprise 製品のスペシャリストがカテゴリを管理できます

Motorprise の顧客には、それぞれに固有のビジネス要件を満たすカタログを提供する必要があります。Motorprise が販売するすべての製品をすべての顧客が購入するわけではありません。Motorprise は、カタログ内の品目と価格を定める契約の交渉を顧客と行います。

最初に、ATG Control Center の「カタログ管理」→「カタログ」セクションで、Motorprise が販売する在庫品のカテゴリごとにサブカタログが作成されました。

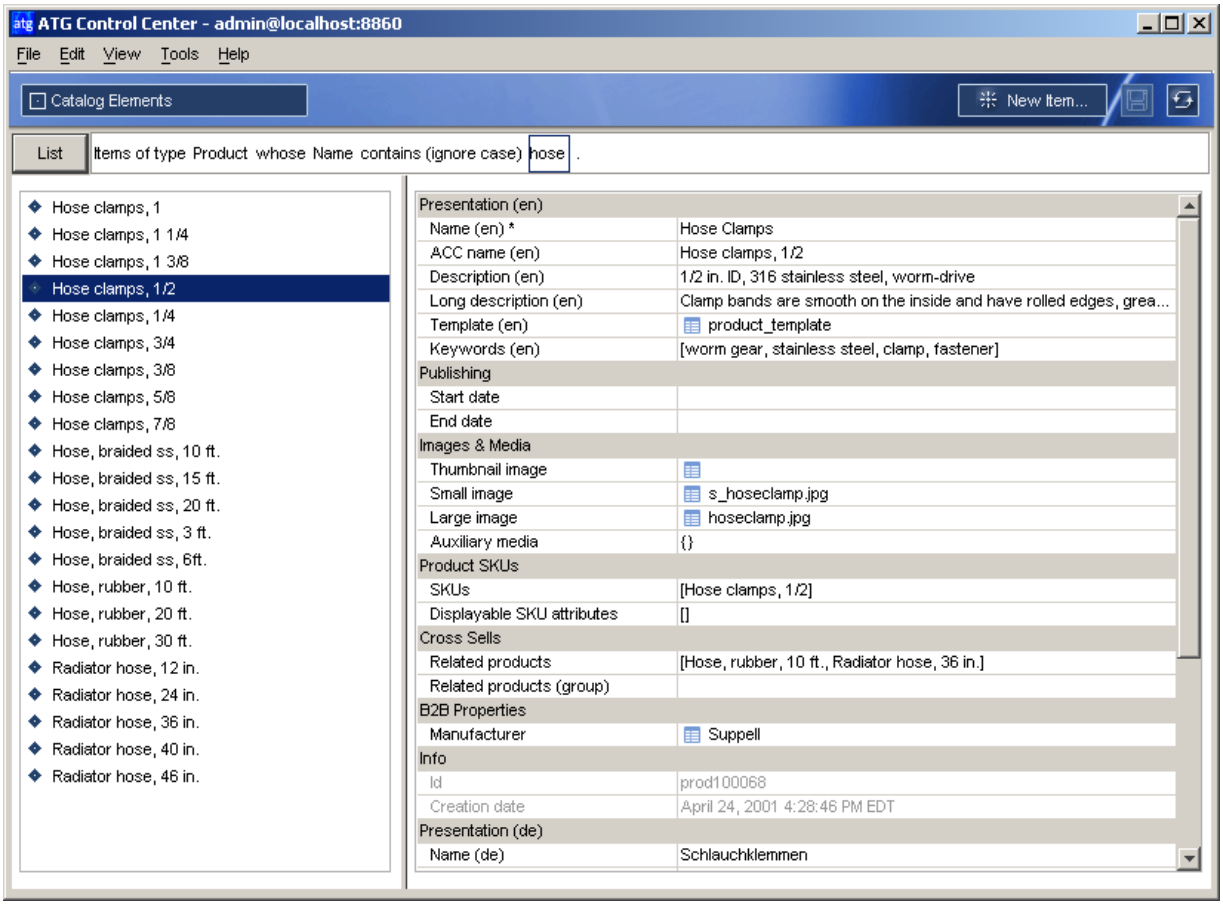
次に、それらのサブカタログから各顧客用のカタログが作成されました。最後に、すべてのサブカタログが含まれるベース・カタログが作成されました。匿名ユーザーもサイトを参照できます。ベース・カタログはデフォルトであり、カタログが割り当てられていない匿名ユーザーにはベース・カタログが表示されます。

これらのビジネス・ニーズは Motorprise 固有のものです。Core Commerce の柔軟なカタログ管理機能を使用して、各自のビジネス戦略に特に適したカタログ構造を作成および管理できます。



ACC におけるカタログ構造の設定

ATG Control Center の「カタログ管理」→「カタログ要素」セクションでは、カテゴリ、製品、SKU など、タイプ別に項目を表示でき、そのプロパティを参照できます。問合せエディタを使用して、正確な項目を簡単に検索することもできます。



「カタログ管理」→「カタログ要素」でのホース製品の参照

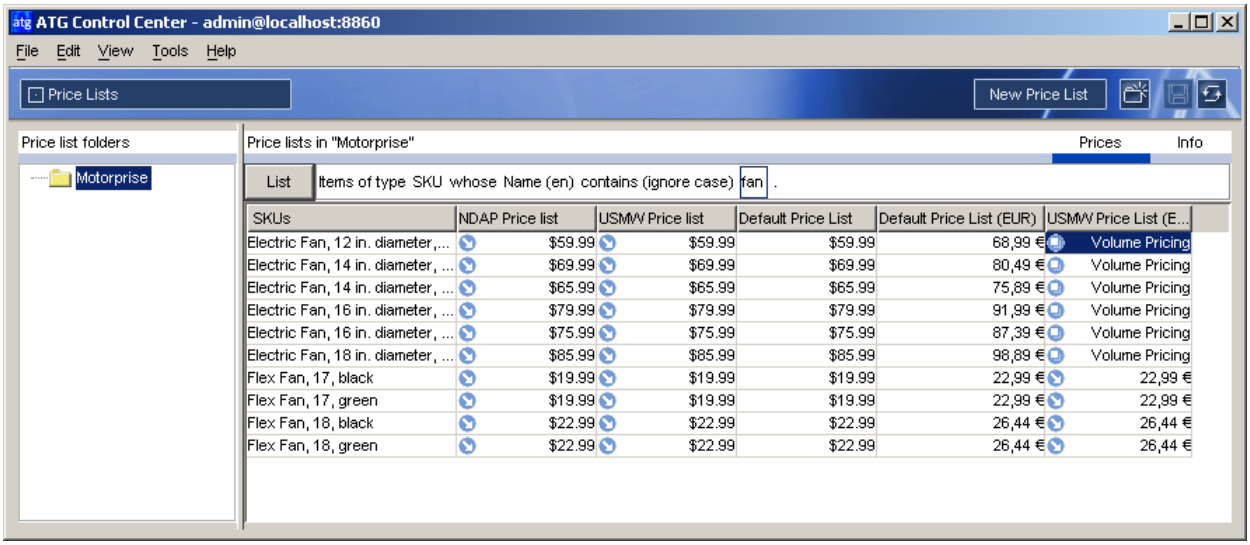
Motorprise カタログは、標準の Core Commerce 機能を拡張して作成されています。詳細は、『ATG Web Commerce Programming Guide』を参照してください。

価格設定

Motorprise では、数量価格設定を伴う価格表が実装されています。Motorprise は B2B サイトのため、顧客は大量のオーダーを頻繁に発行し、顧客との関係に基づいて特別な価格設定の交渉が頻繁に行われます。

各 Motorprise 顧客にカスタムの価格設定を提供する必要があります。Core Commerce には、カスタム価格表を設定する機能が用意されています。顧客の価格表に他に価格がない場合に使用されるデフォルトの価格表が作成されています。また、顧客が特別価格を管理するためのカスタム価格表が設定されています。

Core Commerce には、2 種類の数量価格設定（一括価格設定と段階的価格設定）があります。一括価格設定の場合、最低オーダー量に基づいて製品の価格が計算されます。段階的価格設定の場合は、異なる価格設定レベルの定量または固定重量を使用して製品の価格が計算されます。



Motorprise 価格表の表示

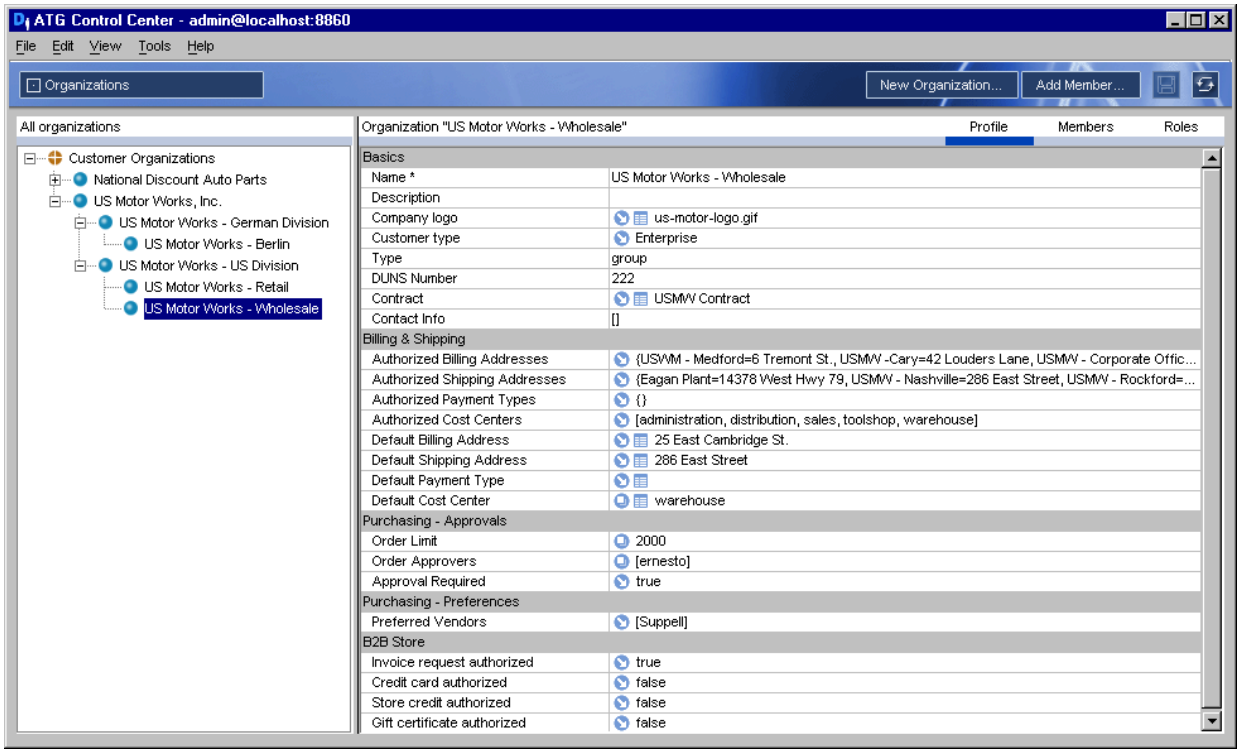
組織階層、プロファイルおよびロールの作成

Motorprise は B2C ではなく B2B サイトのため、個別の各ユーザーだけでなく、各顧客組織とその各サブグループに関する情報も追跡して記録する必要があります。

Motorprise では、複数の層からなる組織構造が作成されています。

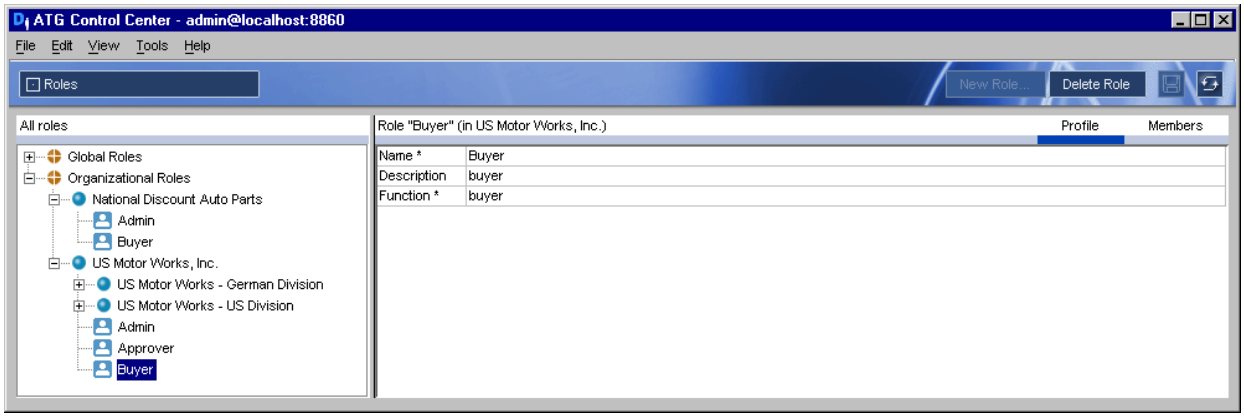
- 会社組織全体
- 部署/ビジネス・ユニット
- グループ（通常は部門）
- 個人

「Motorprise のユーザー・プロファイルおよび組織プロファイルの定義」に、その詳細が記載されています。



ATG Control Center における Motorprise 顧客組織

Motorprise には、バイヤー、承認者および管理者用の様々な顧客ロールも作成されています。購買組織の個人にはロールが割り当てられ、親組織に関連付けられます。個人は親組織の様々なプロパティを継承します。ATG Control Center の「個人および組織」→「ユーザー」セクションで、個々のプロファイルを表示できます。



ATG Control Center における個々のロールの表示

ATG Control Center グループおよびロールの作成

Motorprise ユーザー、つまり、特定のアクセス権限を持つ Motorprise 従業員の、ATG Control Center グループおよびロールも作成されています。たとえば、顧客のカタログおよび価格表の編集権限ならびにベース・カタログおよびデフォルト価格表の読取り専用権限を持つ Motorprise アカウント・マネージャのグループが作成されています。

オーダー処理

Motorprise 顧客には、オーダー処理の際に様々なオプションが提供されます。

- 後でオーダーを保存するオプション
- 定期オーダーをスケジュールする機能
- 購買オーダー、購買依頼、クレジット・カードなど、支払方法の選択
- 明細品目ごとの複数の支払方法、出荷先所在地およびコスト・センターのサポート

3 購買組織としての Motorprise サイトの操作

Motorprise は架空の自動車部品販売業者です。Motorprise Web サイトは、自動車メーカー、自動車販売店、その他の販売業者など、ビジネス顧客に対処するために作成されました。

Motorprise Web サイトは様々な方法で操作できます。

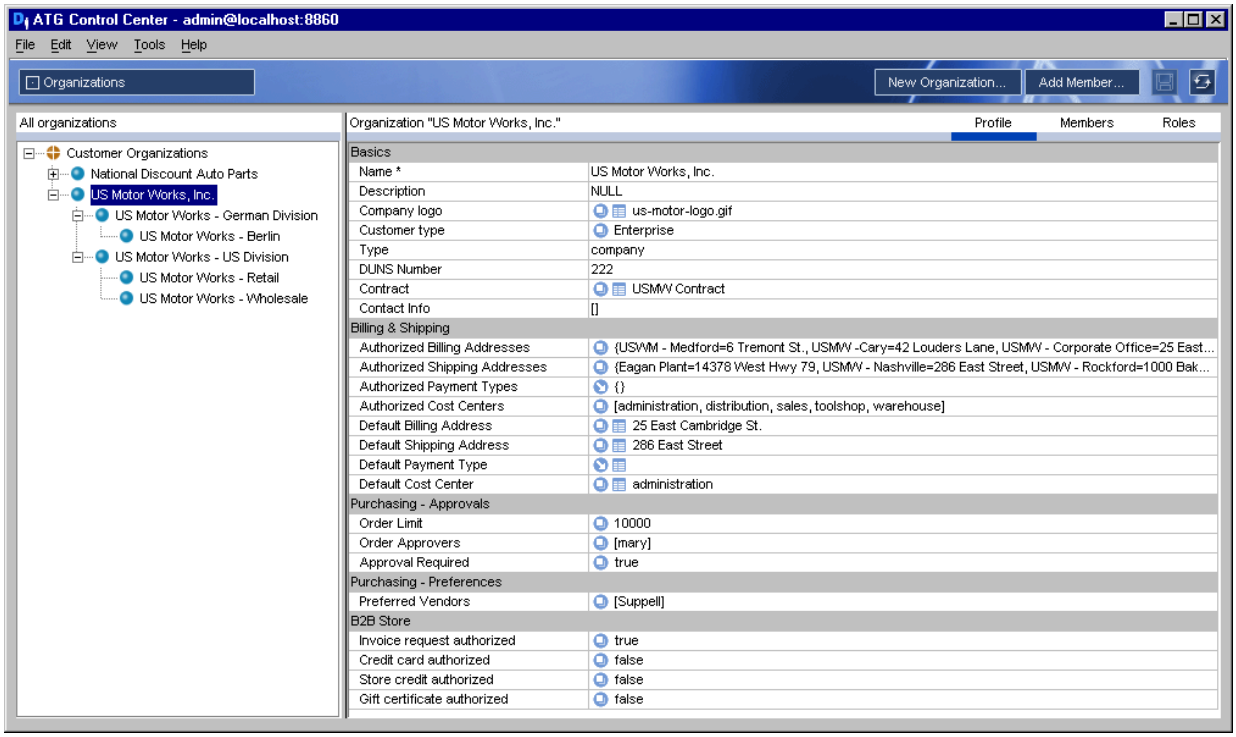
Motorprise 顧客として Web サイトを操作できます。Motorprise リファレンス・アプリケーションには、2 種類の購買組織およびユーザーが用意されています。これらのビジネス・エンティティには、異なる部署レベルと異なるロール（バイヤー、承認者、管理者など）が含まれています。ユーザーとしてログインし、ビジネス・ニーズに対してサイトがどのようにカスタマイズされているかを体験できます。

ATG Control Center に開発者としてログインしてサイトの構築に使用された JSP とコンポーネントを調べ、カタログ、ユーザー・プロファイル、組織プロファイル、シナリオおよびレポート・ツールを表示および変更できます。

Motorprise 管理者として ATG Control Center にログインして、Motorprise を操作することもできます。詳細は、このマニュアルの「販売組織としての Motorprise の操作」を参照してください。

Motorprise 購買組織

Oracle Commerce Core Commerce を使用して多様な顧客のニーズを満たす B2B コマース・サイトを構築する方法を示すために、Motorprise リファレンス・アプリケーションには 2 種類の購買組織が作成されています。ATG Control Center の「個人および組織」→「ロール」セクションで、Motorprise 顧客の組織構造を確認できます。



ATG Control Center における Motorprise 組織プロファイル

National Discount Auto Parts

National Discount Auto Parts（NDAP）は自動車部品を販売する全国的な小売店チェーンです。Motorprise ではこれを優先顧客と分類しています。

購買プロファイル

NDAP はコスト意識は強いですが、顧客に対して優れた顧客サービスを提供することに大きな関心を寄せています。たとえば、必要なときに製品の在庫があることが優先されると考えています。

NDAP ユーザー

ユーザー	組織	ロール
Stuart Lee	National Discount Auto Parts	管理者 バイヤー
Louis Veloso	National Discount Auto Parts	バイヤー

US Motor Works, Inc.

US Motor Works, Inc.（USMW）は大手の国際的な自動車部品販売業者です。USMW はグローバル企業です。Motorprise ではこれを企業顧客と分類しています。

購買プロファイル

Motorprise は USMW を最も戦略的な顧客の 1 つであると見なしています。Motorprise サイトは、USMW ユーザーが実際の在庫量を確認できるように設計されています。他の顧客は在庫状態のみ確認できます。

USMW には複数の部署があります。このリファレンス・アプリケーションでは、米国とドイツの 2 つの部署が例として使用されます。米国の部署には Wholesale グループと Retail グループがあり、ドイツの部署には Berlin グループがあります。

NDAP ユーザー

ユーザー	組織	ロール
Mary Granger	USMW, Inc.（企業レベル）	管理者 承認者 バイヤー
Tim Hartwell	USMW, Inc.（企業レベル）	バイヤー
Ernesto Hernandez	US Division - Wholesale	承認者 バイヤー
Meredith Chin	US Division - Wholesale	バイヤー
Ron Blooming	US Division - Wholesale	バイヤー
Nicole Hsu	US Division - Wholesale	バイヤー
Blair Parrish	US Division - Retail	承認者 バイヤー
Lorna Perman	US Division - Retail	バイヤー
Peter Grün	US Motor Works – German Division	管理者 承認者 バイヤー
Udo Bauer	US Motor Works – German Division US Motor Works – Berlin	承認者 バイヤー
Ingrid Hünä	US Motor Works – German Division US Motor Works – Berlin	バイヤー

バイヤーとしての Motorprise の操作

Motorprise Web サイトにログインするには、次の URL を使用します。

```
http://hostname:port/Motorprise
```

使用するポートは、アプリケーション・サーバーとその構成方法に依存します。たとえば、デフォルトでは、Motorprise の JBoss URL は次のとおりです。

```
http://hostname:8080/Motorprise
```

ログイン・ページで、任意のユーザーの名をユーザー名およびパスワードとして入力することによって、Motorprise ユーザーとしてログインします。

（**注意:** admin は、Motorprise の有効なユーザー名およびパスワードではありません）

Tim Hartwell は Motorprise の登録顧客です。Tim は企業レベルの集中買付けバイヤーとして US Motor Works に勤めています。

ユーザー名とパスワードに tim を使用して、Tim Hartwell としてログインします。

Tim のページはパーソナライズされており、かつ共同ブランド化されています。画面の右上のログイン情報の上に、US Motor Works ロゴが表示されます。USMW が指定する優先ベンダーからの特別提供品が右側に表示されます。これらのベンダーは USMW の組織プロファイルで定義されます。マーチャンダイジング・シナリオ（PreferredSupplierSpecials）によって、ホーム・ページ上でこれらのベンダーの製品がハイライトされます。

Cooling System、Brakes、Ignition and Tune-Up、Tools、Exhaust、Air and Fuel および Electrical というカテゴリを持つ USMW のカタログも表示されます。ページ中央の販促のターゲットは Tim です。オープン・オーダー、履行済オーダーおよび否認済オーダーのステータスも表示されます。

「My Account」をクリックします。

「My Account」の操作

「My Account」ページには、ユーザーのセルフサービス領域があります。彼のこのページには多くのオプションがあります。Tim は次の操作を行うことができます。

- オープン・オーダーと履行済オーダーのステータスを確認できます
- 購入リスト（頻繁に購入される製品のリスト）を格納および作成できます
- 将来のオーダー（1 回かぎりおよび定期的）をスケジュールできます
- 保存されたオーダーを確認できます
- プロファイル情報を編集できます

Tim Hartwell の「My Account」ページ

オーダーの表示

「**Open orders**」リンクをクリックします。次に、特定のオーダー番号のリンクをクリックします。

オーダー全体の詳細が表示されます。このページからそれを取り消すこともできます。

ページ上部のブレッドクラム「My Account」→「Open Orders」→「Order ########」をクリックして、「My Account」ページに戻ります。

履行済オーダーおよび否認済オーダーも同様にレビューできます。

このページから、定期オーダーの表示と作成、購入リストの作成、およびオーダーの保存を行うこともできます。これらの機能は、カタログの参照時や精算時など、サイトの別の領域でも使用できます。

プロファイル情報の編集

「**My profile**」リンクをクリックして、Tim のプロファイルに移動します。

Tim のプロファイルには、ユーザー・プロファイルに格納されている個人の連絡先情報と、USMW とそのビジネス・ユニットの組織プロファイルに格納されている会社情報が含まれています。会社情報には、ビジネス・ユニット、コスト・センター、出荷先所在地などの属性が含まれています。ユーザーは親組織からこのようなプロファイル属性を継承できます。

Motorprise サイトでは、Tim は個人の連絡先情報を編集することや、パスワードを変更することができます。会社情報の編集は、コスト・センターおよび出荷先所在地のリストからのデフォルトの選択に限られます。会社の管理者ではないため、会社情報の追加または編集を行うことはできません。

Tim 宛の E メールを受信するには、Tim のプロファイル内の E メール・アドレスを自分のアドレスに変更します。この操作は任意のユーザーに対して行うことができます。

カタログの参照

「Product Catalog」をクリックします。このカタログは、US Motor Works 用に特別に作成されたものです。左パネルには、USMW カタログ内の 7 つのサブカタログへのリンクが表示されます。「Air and Fuel」カテゴリをクリックします。「Air and Fuel」内には、「Filters」、「Sensors」、「Fuel Pumps」、「PCV Valves」および「Carburetors」という異なるサブカテゴリがあります。

「Fuel Pumps」をクリックし、先頭の「Electric Fuel Pump, part #KAW-1761」をクリックします。

このページには、価格、製造業者、説明など、製品の情報が表示されます。製品の可用性ステータスは在庫の有無を表し、在庫レベルは実際に使用可能な製品数を表します。USMW は企業顧客のため、Tim にはこの製品の在庫レベルが実際に表示されます（優先顧客の National Discount Auto Parts の Stuart Lee というユーザーとしてログインすると、使用可能な実際の数量ではなく、在庫状況のみが表示されます）。

製品ページでは、Tim は様々な操作を実行できます。

- この製品を 1 つ以上、オーダーに追加できます。
- この製品を 1 つ以上、購入リストに追加できます。
- 新規購入リストを作成できます。
- この製品を比較リストに追加できます。
- 関連品目を表示できます。

購入リストの操作

Tim は、自分の購入リストの 1 つに品目を追加することや、新規購入リストを作成することができます。頻繁にオーダーする品目グループの購入リストを作成することで、オーダーするたびに製品カタログから該当の各品目を追加する手間が省けます。

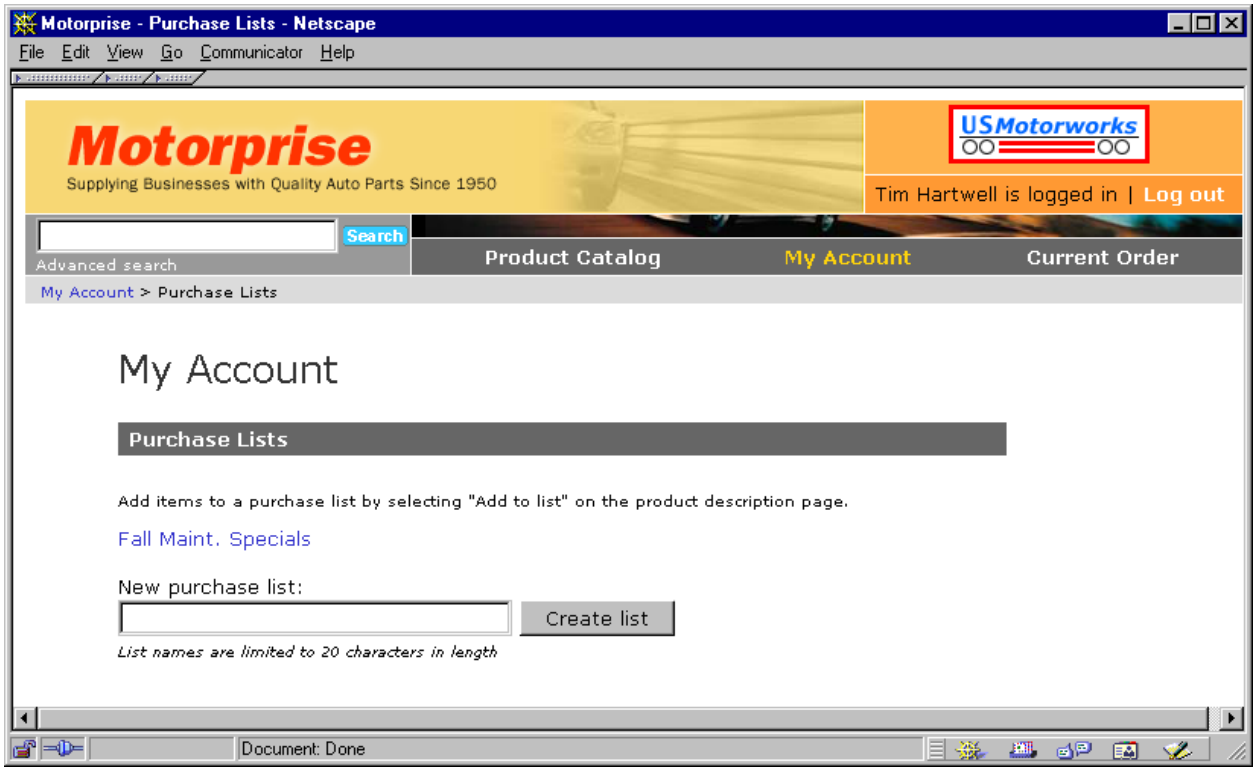
購入リストへの品目の追加

以前に作成したリストをプルダウン・メニューから選択し、**「Add to list」**ボタンをクリックすることで、購入リストの 1 つに任意の個数の品目を追加できます。品目は自動的に追加され、製品カタログでの操作を続けることができます。

購入リストの作成

対応するリンクをクリックすると「My Account」に「Purchase Lists」ページが表示され、新規購入リストを作成できます。ここで、購入リストの追加または変更を行うことができます。購入リストは複数のユーザー・セッションにわたって保持されるため、バイヤーは現在のオーダーに何度も品目を追加できます。

燃料ポンプの製品ページに戻るには、ブラウザの戻るボタンをクリックするか、ヘッダーのブレッドクラム（「Product Catalog」→「Air and Fuel」→「Fuel Pumps」→「Electric Fuel Pump」）を使用します。



「My Account」での購入リストの管理（カタログに戻るにはブレッドクラムを使用）

オーダーの発行

「Add to Order」において、数量フィールドに「120」と入力し、**「Add to order」**ボタンをクリックすることによって、120 単位の KAW-1761 燃料ポンプを Tim のオーダーに追加します。

「Current Order」ページでは、バイヤーが後で作業するためにオーダーを保存すること、品目の数量を変更してオーダーを更新すること、または精算することができます。

「Checkout」ボタンをクリックします。

出荷

「Shipping」ページでは、バイヤーが所在地を選択することや、複数の所在地へ出荷することができます。Tim は、複数の小売店用の購買を行う集中買付けバイヤーです。単一の大量オーダーを発行し、その一部を複数の所在地へ出荷できます。

「Ship to multiple addresses」リンクをクリックします。

「Quantity to move」フィールドに「60」と入力し、ドロップダウン・リストから別の所在地（US Motor Works – Nashville）を選択して、オーダーを分割します。ドロップダウン・リストには、様々な出荷先所在地のニックネームが表示されています。**「Save」**ボタンをクリックします。オーダーが 2 つの所在地に分割されます。**「Continue」**ボタンをクリックします。

各出荷先グループには、それぞれの出荷方法があります。ドロップダウン・リストを使用して各所在地の出荷方法を選択します。**「Continue」**ボタンをクリックします。

請求

「Billing」ページでは、支払方法を入力します。Tim は USMW バイヤーとして、購買オーダーまたは購買依頼の使用が許可されています（クレジット・カードの使用は許可されていません）。1 つのオーダーに対して複数の支払方法を使用でき、金額や明細品目でオーダーを分割できます。後で、オーダーの各部にコスト・センターを割り当てることもできます。

「Split order by dollar amount」リンクをクリックします。P.O.番号を入力し、**「Add」**をクリックします。Tim は必要に応じてこの P.O.番号の請求先所在地を変更できることに注意してください。続けて、別の P.O.番号を追加します。複数の P.O.番号を入力すると、ページ下部のドロップダウン・リストから選択することによって、それをデフォルトの支払方法として指定できます。

P.O.番号の入力が終了したら、**「Continue」**ボタンをクリックします。オーダーの全額がデフォルト P.O.に割り当てられています。ここで、移動する金額を入力し、P.O.を選択し、**「Save」**を選択することによって、特定の金額を各購買オーダーに割り当てることができます。

異なる P.O.番号に金額を移動し、**「Save」**をクリックします。**「Continue」**ボタンをクリックします。

会計処理を円滑に進めるために、複数のコスト・センターをオーダー内の各明細品目に割り当てることができます。この機能は、複数の支払方法と同様に動作します。必要に応じて複数のコスト・センターを選択し、**「Continue」**をクリックします。

オーダーの確認ページで、定期オーダーの作成、オーダーの即時実行またはオーダーの取消しを行うことができます。**「Place order」**をクリックします。

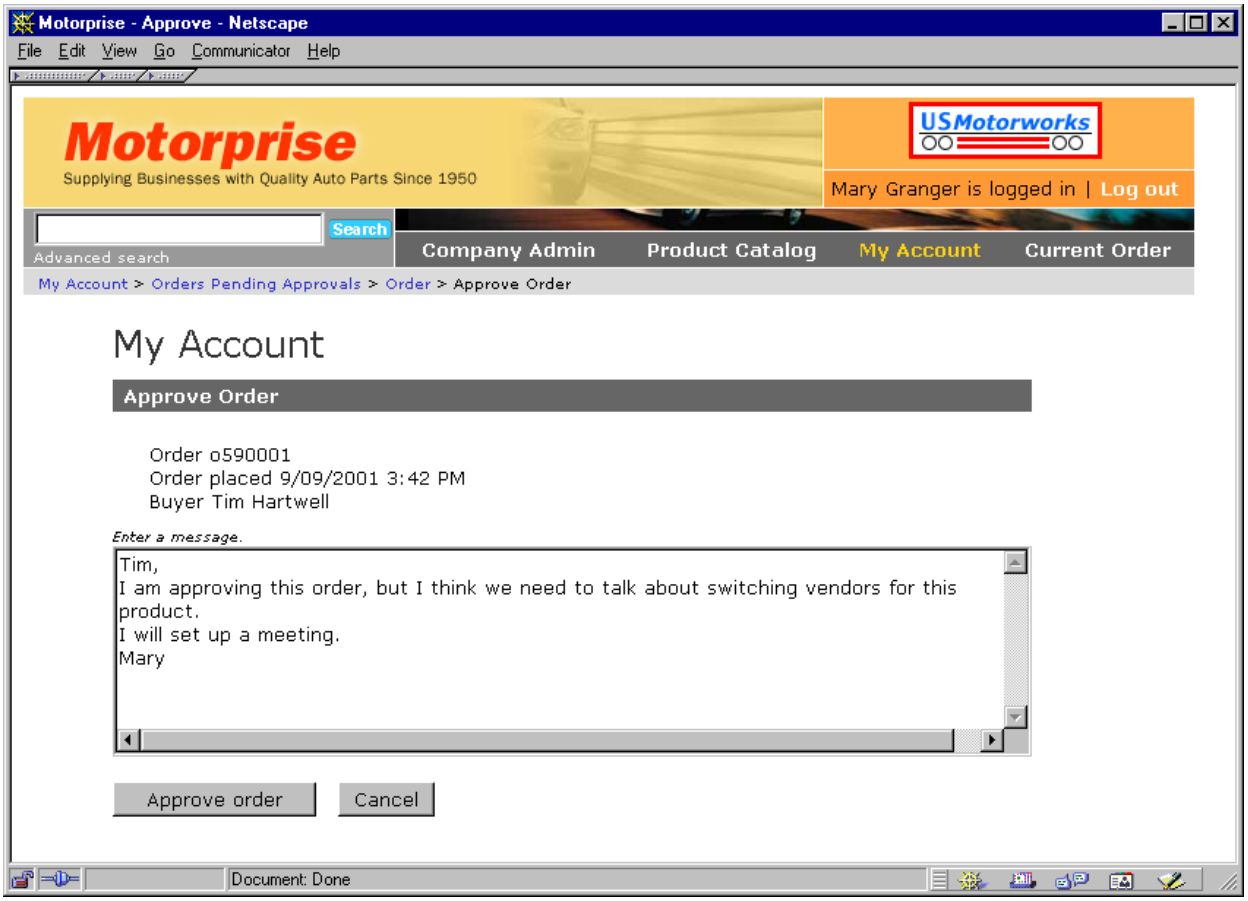
Tim に許可されているオーダーの限度額は$10,000 です。それを超えるすべてのオーダーについて、スーパーバイザの Mary Granger からオーダー承認を受ける必要があります。Tim のオーダーが承認を求めて発行されました。承認が必要なオーダーがあることを Mary に E メールで通知するように、ApprovalNotification シナリオが設定されています。Mary がこの E メールを受信すると、ログインしてオーダーを確認します。

承認者としての Motorprise の操作

「Log out」をクリックし、ユーザー名およびパスワードに mary を使用して Mary Granger としてログインします。

Mary は会社の承認者であり、管理者です（Tim の場合は表示されなかった「Company Admin」領域が表示されます）。彼女のホーム・ページには、承認が必要なオーダーが表示されます。**「Approvals」**リンクをクリックします。

Mary の承認が必要なすべてのオーダーが表示されます。Tim が発行したオーダーをクリックします。Mary は保留中のオーダーをレビューし、それを承認または否認できます。**「Approve order」**リンクをクリックします。Tim に通知する前に、オーダーにメッセージを追加できます。

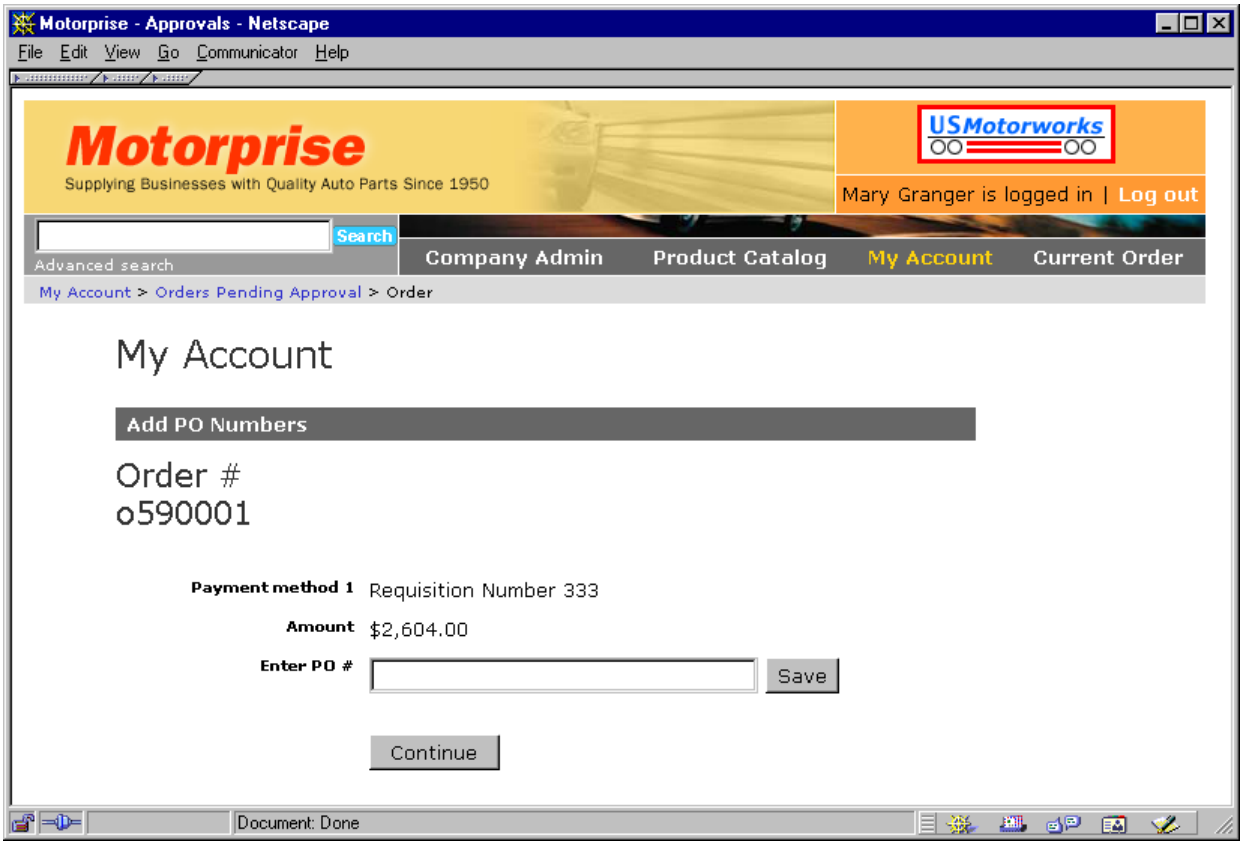


承認者はオーダーの否認または承認時にメッセージを入力できる

オーダーに対するコメントを入力し、「**Approve order**」ボタンをクリックします。Mary がオーダーを承認すると、これが Motorprise に発行されて履行され、Tim にオーダーの承認/否認ステータスを通知する別のシナリオがトリガーされます。

「My Account」をクリックします。Mary は承認者であるため、ここで現在および過去の承認情報を表示できます。

ユーザーが発行したオーダーに承認が必要で、そのオーダーには購買依頼番号はあるが P.O.番号がない場合、Motorprise ではクレジット・カード以外のオーダーの場合に P.O.番号が必要なため、その追加が Mary に対して求められます。



バイヤーが承認を求めて発行したオーダーに購買依頼番号のみがある場合、承認者は P.O.番号を入力する必要がある。

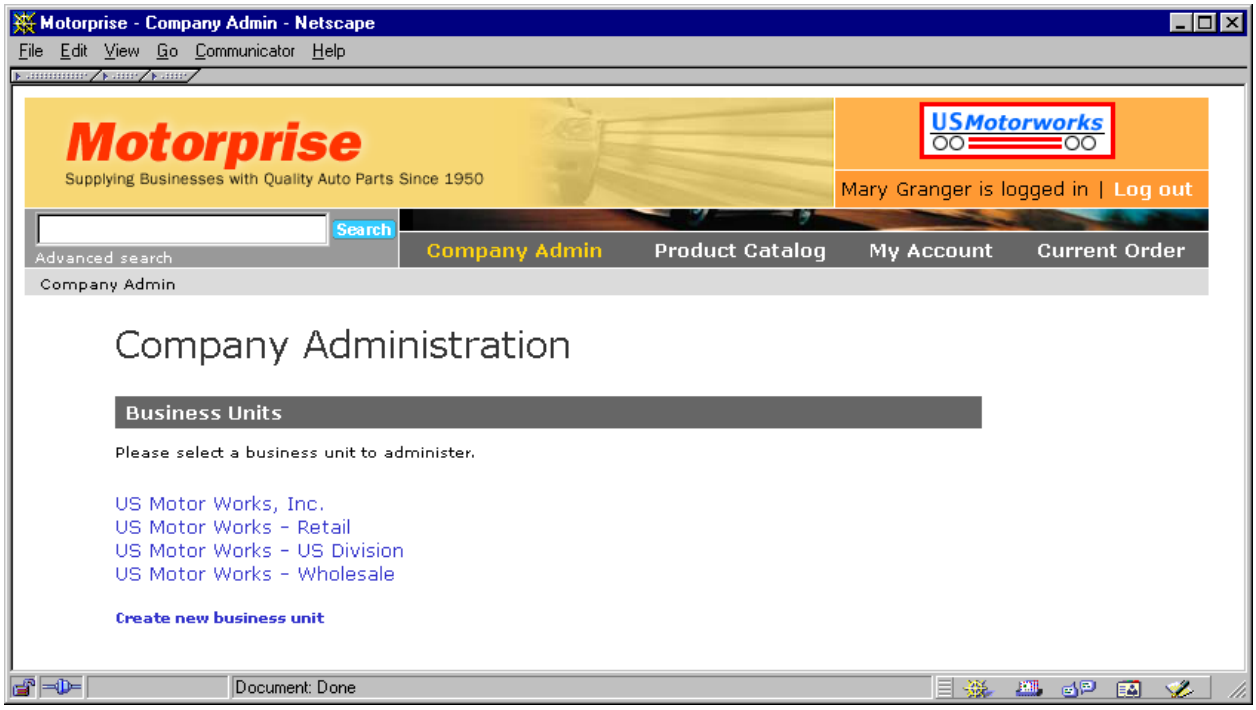
会社の管理者としての Motorprise の操作

Motorprise では、会社管理用の JSP インタフェースを通じて、購買組織が各自のユーザー・プロファイルと組織プロファイルを管理できます。この機能により、購買組織は、供給組織（この場合は Motorprise）に依存せずに、Web ブラウザを通じて、組織の情報およびロールを管理できます。

Mary は管理者のため、Motorprise サイトにログインすると「Company Admin」領域にアクセスできます。「Company Admin」セクションをクリックします。

ページには、Mary が管理者権限を持つすべてのビジネス・ユニットがリストされます。彼女は US Motor Works, Inc.の企業レベルの管理者であるため、米国の部署全体と、その階層に属するすべてのグループにアクセスできます。このページから新しいビジネス・ユニットを作成することもできます。

ビジネス・ユニットのリストから 1 つのビジネス・ユニット（たとえば、「US Motor Works – Wholesale」）を選択します。



管理者は「Company Admin」ページで組織のビジネス・ユニットから選択する。

ビジネス・ユニットを選択すると、ユーザー・プロファイルを作成および編集でき、ユーザーを特定のロールに関連付けることができます。複数のユーザー・プロファイルを同時に編集および追加することもできます。管理者は、所在地、支払方法、コスト・センター、購入限度などのビジネス・ユニット情報も編集できます。

管理者オプション

複数のユーザーの作成

「**Create Multiple Users**」リンクをクリックします。作成するユーザー数を入力し、「create users」を選択します。ここでは、各ユーザーの個別情報（名、ミドル・ネーム、姓、ログイン、パスワード、E メール・アドレスなど）を入力できます。使用可能なロールを各ユーザーに割り当てることができます。Motorprise でビジネス・ユニットを作成すると、デフォルトでバイヤー、承認者および管理者の各ロールが含まれます。

新規ユーザーを作成するには、「**Save**」ボタンをクリックします。「Company Admin」ページに戻るには、「**Cancel**」ボタンをクリックします。NewUser シナリオ（有効な場合）によって、新規ユーザーにユーザー名とサイトへのリンクを伝える E メールが送信されます。

複数のユーザーの更新

「**Edit multiple users**」リンクをクリックします。更新するユーザーを選択します。ここでは、編集する属性をリストから選択し、各属性の値を選択できます。「**Change for selected users**」ボタンをクリックします。確認画面から、ユーザー名をクリックして個々のユーザー・プロファイルを編集できます。

出荷先所在地の管理

「**Shipping addresses**」リンクをクリックします。

「Company Admin」内の「Shipping Address」ページでは、出荷先所在地を作成、変更および削除できる。

新規出荷先所在地を作成するには、**「Create new shipping address」**リンクをクリックします。新規出荷先所在地の情報を入力し、**「Save」**ボタンをクリックします。

変更する出荷先所在地の**「Edit」**ボタンをクリックすると、既存の出荷先所在地を編集できます。新しい情報を入力し、**「Save」**ボタンをクリックします。削除する出荷先所在地の**「Delete」**ボタンをクリックすると、出荷先所在地を削除できます。

請求先所在地の管理

出荷先所在地の場合と同様に、Mary は請求先所在地を作成、編集および削除できます。

コスト・センターの管理

「Cost centers」リンクをクリックします。

新規コスト・センターを作成するには、**「Add new cost center」**リンクを選択します。コスト・センターの ID 番号や名前などの情報を入力できます。**「Save」**ボタンをクリックします。該当のリンクをクリックして、既存のコスト・センターを編集または削除できます。

承認

ここでは、このビジネス・ユニットに承認が必要かどうか、および必要な場合は、その購入限度を指定できます。

支払方法の許可

「**Authorize payment methods**」リンクを選択します。

ビジネス・ユニットに請求書、クレジット・カードまたはその両方の使用が許可されているかどうかを指定できます。

4 Motorprise のユーザー・プロファイルおよび組織プロファイルの定義

この章では、Motorprise 用に標準のユーザー・プロファイルをどのように拡張したかについて説明します。この章は次の項から構成されています。

プロファイル・リポジトリの拡張
Motorprise 固有のプロパティを `user` 項目タイプおよび `organization` 項目タイプに追加することによって、プロファイル・リポジトリを拡張した方法について説明します。

テーブル定義の作成
ユーザー・プロファイル・データを格納するデータベース・スキーマを、SQL スクリプトを使用して作成した方法について説明します。

プロファイル・リポジトリの拡張

この項では、Motorprise サイト用に Oracle Commerce Core Commerce および Oracle Commerce Personalization に対して行った拡張について説明します。Oracle Commerce Personalization は、Motorprise サイトの個々のユーザー固有のデータの収集、格納および取得を行います。プロファイルは、ATG Control Center に入力された情報に基づき、または Motorprise サイト上の会社管理者やユーザーによる処理によって、個々の顧客または組織用に格納されるデータです。このデータを収集することによって、Web サイト開発者、管理者およびビジネス・ユーザーは、顧客に対して、そのプロファイル情報に基づいてパーソナライズされた製品とコンテンツを表示できます。

Motorprise では、様々な形でパーソナライズを使用して、強力な B2B 機能が実現されています。Oracle Commerce Personalization は基本のパーソナライズ機能を提供し、Core Commerce はこの機能を B2B 固有のプロパティで拡張します。自分の Web アプリケーションのニーズに合うように、この機能は簡単に調整できます。Motorprise はこの機能を拡張する様々な方法を示しています。

SQL リポジトリ定義の詳細は、『ATG Web Commerce Personalization Programming Guide』の「*プロファイル・リポジトリの設定*」および『ATG Web Commerce Repository Guide』の「*SQL コンテンツ・リポジトリ*」を参照してください。

Oracle Commerce Personalization は、`user` 項目タイプ、`organization` 項目タイプおよびその他のサポートする項目タイプで構成されるリポジトリに、顧客データを格納します。デフォルトの Oracle Commerce Personalization リポジトリには、名前や所在地の情報など、多くのパーソナライズされた Web アプリケーションで必要な一連のプロパティを持つ項目タイプが含まれています。Core Commerce では、`user` 項目タイプおよび `organization` 項目タイプにプロパティを追加することで、多くの B2B 機能が実現されます。次に説明するように、Core Commerce の上に Motorprise レベルでプロパティが追加されています。

Motorprise プロパティの追加

Motorprise では、ビジネス・モデルに固有の機能を実現するために、user および organization の項目タイプにプロパティが追加されています。これらの拡張機能は、<ATG11dir>/Motorprise/config/atg/userprofiling/userProfile.xml 内で定義されています。

Motorprise の organization リポジトリ項目および user リポジトリ項目内のプロパティの多くは、他のプロパティから導出されます。導出プロパティにより、あるリポジトリ項目が、別のリポジトリ項目または同じリポジトリ項目内の別のプロパティから値を取得できます。導出プロパティの詳細は、『ATG Web Commerce Platform Programming Guide』を参照してください。

たとえば、<ATG11dir>/Motorprise/config/atg/userprofiling/userProfile.xml 内の次の XML コードは、Motorprise 内のプロパティ invoiceRequestAuthorized の導出プロパティの使用方法を示しています。

一部の Motorprise ユーザーは、購買オーダーと請求書要求の使用は承認されていますが、会社のクレジット・カードの使用は許可されていません。ユーザーまたは組織がその承認を受けているかどうかを確認するために、Motorprise 内に特定のプロパティが追加されています。

まず、organization 項目記述子内に導出プロパティ invoiceRequestAuthorized が作成されています。

```
<item-descriptor name="organization">
    <table name="b2b_org_info" type="auxiliary" id-column-name="org_id">
      <property category-resource="categoryB2BStore"
        name="myInvoiceRequestAuthorized"  data-type="boolean"
        column-name="invoice_auth"
        display-name-resource="myInvoiceRequestAuthorized" expert="true">
      </property>
    </table>
    <!-- Derived properties -->
    <property category-resource="categoryB2BStore" name="invoiceRequestAuthorized"
      data-type="boolean" display-name-resource="invoiceRequestAuthorized">
      <derivation override-property="myInvoiceRequestAuthorized">
        <expression>parentOrganization.invoiceRequestAuthorized</expression>
      </derivation>
    </property>
</item-descriptor>
```

Motorprise では、すべての組織が、その組織に属するユーザーに購買オーダーの使用が許可されているかどうかを示す、ブール型プロパティ invoiceRequestAuthorized を持ちます。すべての組織がこのプロパティをチェックします。プロパティが存在しない場合は、親組織から値を導出します。前述のコード・サンプルに示すように、その値を格納する追加のプロパティ myInvoiceRequestAuthorized が organization 内に作成されています。また、<derivation override-property="myInvoiceRequestAuthorized">タグを使用して myInvoiceRequestAuthorized 内の値を実際にチェックする、必須プロパティ invoiceRequestAuthorized が作成されています。この値が null の場合は、タグ<expression>parentOrganization.invoiceRequestAuthorized</expression>を使用して親組織から値を取得します。

次のコードに示すように、organization の invoiceRequestAuthorized プロパティに基づいて、user 項目記述子内に別の導出プロパティが作成されています。

```
<item-descriptor name="user">
    <table name="b2b_auth_pmnt" type="auxiliary" id-column-name="id">
      <property category-resource="categoryB2BStore"
        name="myInvoiceRequestAuthorized" data-type="boolean"
        column-name="invoice_auth" expert="true"
        display-name-resource="myInvoiceRequestAuthorized">
      </property>
    </table>
    <!--Derived Property -->
    <property category-resource="categoryB2BStore" name="invoiceRequestAuthorized"
      data-type="boolean" display-name-resource="invoiceRequestAuthorized">
      <derivation override-property="myInvoiceRequestAuthorized">
        <expression>parentOrganization.invoiceRequestAuthorized</expression>
      </derivation>
    </property>
</item-descriptor>
```

前述のように、任意のユーザーに対して invoiceRequestAuthorized が取得されると、最初にユーザー・レベルでその値をチェックします。それが null の場合は、親組織から値を取得します。これにより、大部分のユーザーが親組織の設定を継承できると同時に、必要に応じて特定のユーザーのデフォルト動作を上書きできます。

Motorprise では、次に示す他の導出プロパティも作成されています。

これらは organization 項目タイプに追加されたプロパティです。

プロパティ	摘要
companyLogo	会社のロゴを表示するメディア項目。
myCompanyLogo	このプロパティは、companyLogo プロパティを上書きします。ビジネス・ユニットのロゴが親組織のロゴと同じでない場合に使用されます。
invoiceRequestAuthorized	支払方法として請求書の使用がユーザーに許可されているかどうかを示します。精算ページはプロファイルのこのプロパティを使用して、オーダーの支払を請求書で行うオプションを表示するかどうかを決定します。
creditCardAUthorized	支払方法としてクレジット・カードの使用がユーザーに許可されているかどうかを示します。
giftCertificateAuthorized	支払方法として商品券の使用がユーザーに許可されているかどうかを示します。
storeCreditAuthorized	支払方法としてストア・クレジットの使用がユーザーに許可されているかどうかを示します。

プロパティ	摘要
myInvoiceRequestAuthorized	これは invoiceRequestAuthorized プロパティの上書きプロパティです。ユーザーが invoiceRequestAuthorized プロパティにアクセスしようとして、user タイプの myInvoiceRequestAuthorized プロパティに値がない場合は、このプロパティがアクセスされます。
myCreditCardAuthorized	このプロパティは myInvoiceRequestAuthorized と同じように動作します。
myGiftCertificateAuthorized	このプロパティは myInvoiceRequestAuthorized と同じように動作します。
myStoreCreditAuthorized	このプロパティは myInvoiceRequestAuthorized と同じように動作します。

また、user 項目タイプには次のプロパティが追加されています。

プロパティ	摘要
currentOrganization	サイトの「Company Administarion」セクションで使用されます。購買組織の管理者（USMW の Mary Granger など）がログインし、ビジネス・ユニットを選択すると、このプロパティが設定されます。このプロパティは organization リポジトリ項目を参照します。他のユーザーの場合、このプロパティは設定されません。
currentUser	user リポジトリ項目への参照です。管理ページで複数のユーザーを操作するときに使用されます。このプロパティは、任意の時点で更新されているユーザーを追跡します。たとえば、管理者 Mary がユーザーStuart のプロパティを編集している場合、currentUser プロパティは Stuart に設定されています。
numOfOrders	ユーザーが発行した現在のオーダー数を示します。このプロパティは 25OverAverage シナリオで使用されます。詳細は、このマニュアルの「マーチャンダイジング」を参照してください。
avgOrderAmt	ユーザーのすべてのオーダーの平均合計額を追跡します。このプロパティは 25OverAverage シナリオで使用されます。詳細は、このマニュアルの「マーチャンダイジング」を参照してください。
invoiceRequestAuthorized	支払方法として請求書の使用がユーザーに許可されているかどうかを示します。精算ページはプロファイルのこのプロパティを使用して、オーダーの支払を請求書で行うオプションを表示するかどうかを決定します。

プロパティ	摘要
creditCardAUthorized	支払方法としてクレジット・カードの使用がユーザーに許可されているかどうかを示します。
giftCertificateAuthorized	支払方法として商品券の使用がユーザーに許可されているかどうかを示します。
storeCreditAuthorized	支払方法としてストア・クレジットの使用がユーザーに許可されているかどうかを示します。
myInvoiceRequestAuthorized	これは invoiceRequestAuthorized プロパティの上書きプロパティです。ユーザーが invoiceRequestAuthorized プロパティにアクセスしようとして、myInvoiceRequestAuthorized プロパティに値がない場合は、親組織のプロパティがアクセスされます。
myCreditCardAuthorized	このプロパティは myInvoiceRequestAuthorized と同じように動作します。
myGiftCertificateAuthorized	このプロパティは myInvoiceRequestAuthorized と同じように動作します。
myStoreCreditAuthorized	このプロパティは myInvoiceRequestAuthorized と同じように動作します。
currentLocation	このプロパティはユーザーが要求するページに設定されます。ユーザーが現在要求しているページの追跡に使用されます。

一時プロパティ currentLocation の設定方法の詳細は、「要求処理パイプライン・サーブレット」の SetCurrentLocation サーブレットに関する項を参照してください。

プロファイル・プロパティの詳細は、『ATG Web Commerce Personalization Programming Guide』の「*Dynamo User Directory を使用しての作業*」を参照してください。

テーブル定義の作成

<ATG11dir>/Motorprise/config/atg/userprofiling/userProfile.xml ファイルは、Oracle Commerce Platform、Scenarios モジュールおよび Oracle Commerce Personalization の userProfile.xml ファイルと結合されます。これらのファイルは XML ルールの組合せごとに結合されて 1 つの XML ファイルを生成し、これが解析および使用されて、リポジトリ内の項目タイプを記述します（XML ファイル結合の詳細は、『ATG Web Commerce Platform Programming Guide』を参照してください。）結合されたファイルはプロファイル・アダプタ・リポジトリが開始するたびに再解析され、ディスクには書き込まれないため、結合されたファイルではなく、個別のファイルのみを保持する必要があります。

この方法により、ユーザーは階層構造でユーザー・プロファイルを定義できます。たとえば、<ATG11dir>/Motorprise/config/atg/userprofiling/userProfile.xml は、Motorprise 固有の該当のプロパティのみを定義します。これは、既存のユーザー・プロファイル定義へのアドオンです。

ユーザー・プロファイル・リポジトリの基礎となる記憶域は、リレーショナル・データベースです。SQL プロファイル・リポジトリを使用して、リポジトリ API を通じてデータを公開します。XML ファイル内では、リレーショナル・データベースのテーブルおよび列へのリポジトリ項目のマッピングを記述します。

ユーザー・プロファイル・データを格納するためのデータベース・スキーマは、次の SQL スクリプトを使用して作成されています。

```
<ATG11dir>Motorprise/sql/db_components/mysql/b2b_user_orddet_ddl.sql
```

プロファイルが拡張されると、ATG Control Center ユーザーは顧客組織用のプロファイルを設定できます。その顧客組織の管理者は、Motorprise Web サイトの「Company Admin」ページを使用して、顧客組織のユーザーおよびサブ組織を作成および編集できます。

5　会社管理

Motorprise Web サイトには、購買組織の管理者がアカウントにリモート・アクセスできる、セルフサービスの「Company Admin」インタフェースが含まれています。

Motorprise 顧客は、サイトの「Company Admin」セクションを使用して、各自のユーザー・プロファイルおよび組織プロファイルを管理できます。管理者は、サブ組織とグループの作成と編集、特定のユーザー・タイプへのユーザーの関連付け、およびユーザー・プロファイルの編集を行うことができます。複数のユーザー・プロファイルを同時に編集および更新することもできます。基本的に、Motorprise 顧客は「Company Admin」インタフェースを通じて、会社のユーザーに関するすべての情報とアクセス権限を管理および制御できます。

この章では、購買組織向けに作成した Motorprise サイトの「Company Admin」セクションについて説明します。この章は次の項から構成されています。

管理者アクセス権限の確認
管理ページへのアクセス権限をユーザーに付与する方法について説明します。

ビジネス・ユニットおよびロールの作成
ビジネス・ユニットおよび対応するロールを作成する方法について説明します。

ビジネス・ユニットの選択
管理するビジネス・ユニットを管理者が選択する方法について説明します。

複数のユーザーの登録
複数のユーザーを同時に作成する方法について説明します。

複数のユーザー・プロファイルの編集
複数のユーザーの共通プロパティを同時に編集する方法について説明します。

ユーザーの削除
ユーザーを削除する方法について説明します。

B2BRepositoryFormHandler の使用
`B2BRepositoryFormHandler` を使用して、管理者による出荷先所在地、請求先所在地、支払方法、コスト・センターおよび購入限度の管理を実現する方法について説明します。

フォーム・ハンドラのスコープの変更
`ultiUserAddFormHandler` および `MultiUserUpdateFormHandler` のスコープを変更する方法について説明します。

管理者アクセス権限の確認

Motorprise では、管理者権限を持つ組織内のユーザーのみが、「Company Admin」ページを表示して使用できます。これらのユーザーには、Motorprise 管理者によってユーザーが作成されるときに admin ロールが

割り当てられています。ユーザーが Motorprise にログインすると、親組織に対する管理者権限を持つかどうかがチェックされます。

ログインしたユーザーが管理者かどうかをチェックするために、ページ・フラグメント MotorpriseJSP/j2ee-apps/motorprise/web-app/en/common/BrandNav.jsp 内の atg/userdirectory/droplet/HasFunction コンポーネントが使用されています。このフラグメントは MotorpriseJSP/j2ee-apps/motorprise/web-app/en/home.jsp で使用されます。

2 つの入力パラメータが HasFunction に渡されます。ログインするユーザーの ID が userId として使用されます。そのユーザーにおける admin の function の有無もチェックされ、ある場合は true の出力が行われます。

BrandNav.jsp からのコード・フラグメントを、次に示します。

```
<!--  display link if user has admin role -->
<dsp:droplet name="HasFunction">
  <dsp:param bean="Profile.id" name="userId"/>
  <dsp:param name="function" value="admin"/>
  <dsp:oparam name="true">
    <dsp:droplet name="Switch">
      <dsp:param bean="Profile.currentLocation" name="value"/>
      <dsp:oparam name="admin">
        <td align="center"><dsp:a href="../admin/business_units.jsp">
        <b><font color="#FDD30E" size=-1>Company
          Admin</font></b></dsp:a></td>
      </dsp:oparam>
      <dsp:oparam name="default">
        <td align="center"><dsp:a href="../admin/business_units.jsp">
        <b><font color="#FFFFFF" size=-1>Company
          Admin</font></b></dsp:a></td>
      </dsp:oparam>
    </dsp:droplet>
  </dsp:oparam>
</dsp:droplet>
```

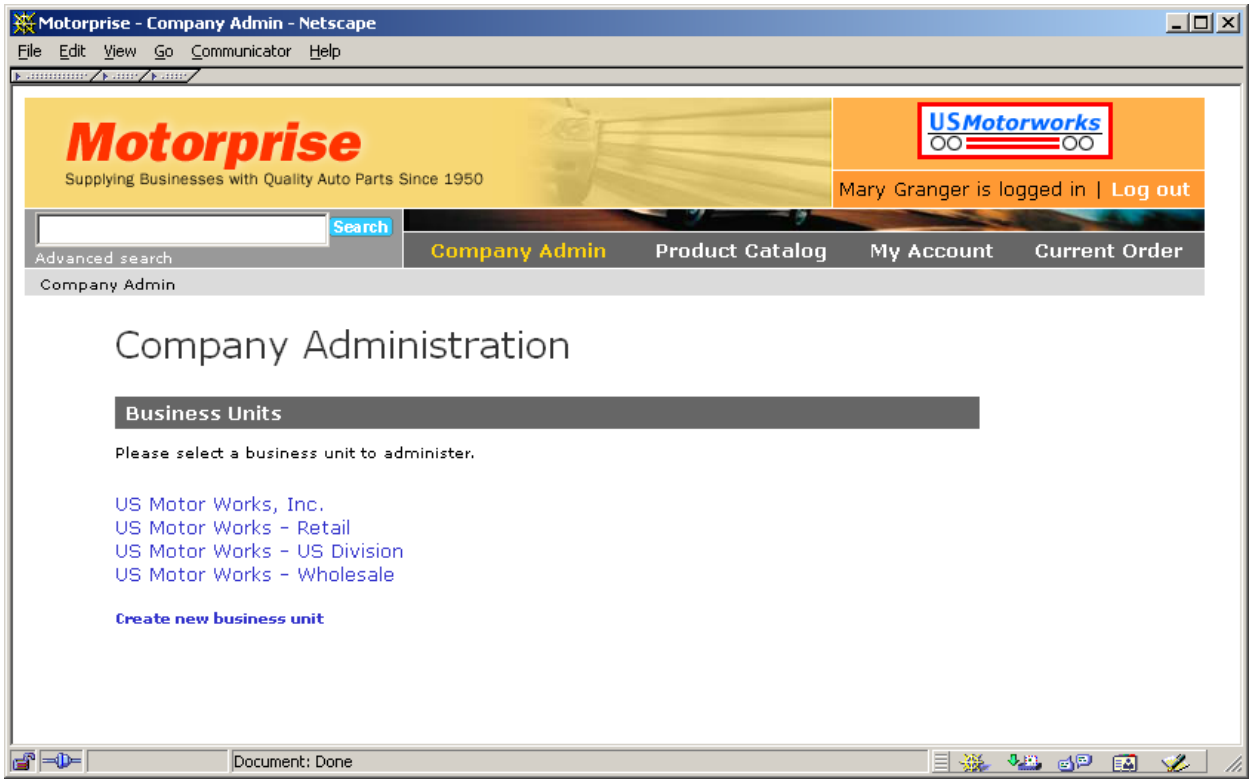
ユーザーが admin の function を持つ場合、ホーム・ページに「Company Admin」リンクが表示され、管理ページへのアクセス権限が付与されます。次の画面ショットは、「Company Admin」リンクがある、USMW の Mary Granger のホーム・ページを示しています。

会社の管理者がログインしたときに表示される「Company Admin」タブ

ユーザーが管理者でない場合は「Company Admin」リンクは表示されず、「Company Admin」ページにアクセスできません。次の画面ショットは、「Company Admin」リンクがない、USMW の Ron Blooming のホーム・ページを示しています。

管理者権限を持たないユーザーがログインしたときには「Company Admin」タブは表示されない

管理者が正しいユーザー名とパスワードでログインし、「Company Admin」を選択すると、「Business Units」ページが表示されます。このページには、管理権限を持つすべてのビジネス・ユニットがリストされます。これらのユニットのいずれかを選択することや、新規ビジネス・ユニットを作成することができます。



USMW の Mary Granger は「Company Administration」ページでビジネス・ユニットの選択または新規作成を行うことができる

ビジネス・ユニットおよびロールの作成

Motorprise では、会社管理者は会社のビジネス・ユニットおよびロールを作成します。次の目的で `atg/projects/b2bstore/userdirectory/CreateOrganizationFormHandler` が使用されます。

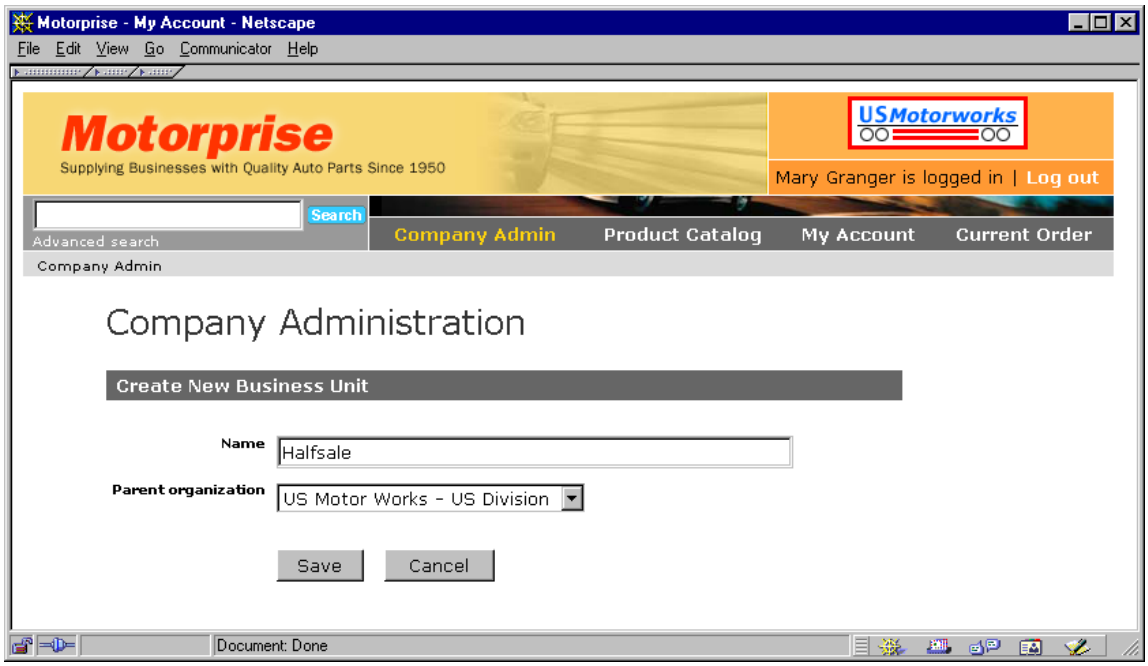
- 新規ビジネス・ユニットの作成
- その名前と親組織の設定
- そのビジネス・ユニットのロール（バイヤー、承認者、管理者など）の作成および割当て
- ビジネス・ユニットを作成したユーザーへのそのビジネス・ユニットの管理者ロールの割当て

Motorprise で管理者が新規ビジネス・ユニットを作成すると、`CreateOrganizationFormHandler` は次の 2 つの処理を行います。

- そのビジネス・ユニットのバイヤー、承認者および管理者に関連するロールの作成
- ビジネス・ユニットを作成したユーザーへの管理者ロールの割当て

新規ビジネス・ユニットの作成

「Company Admin」セクションの「Business Units」ページには、business_unit_new.jsp への「**Create new business unit**」リンクがあります。管理者がこのリンクをクリックすると、「Create New Business Unit」ページが表示されます。



新規ビジネス・ユニットの作成

このページでは、新規ビジネス・ユニットの名前を入力し、既存のビジネス・ユニットを親組織として選択できます。たとえば、Mary Granger は、「Halfsale」というビジネス・ユニットを作成し、「US Motor Works – US Division」を親として選択できます。

新規ビジネス・ユニットを作成し、その名前と親組織を指定するために、<ATG11dir>/MotorpriseJSP/j2ee-apps/motorprise/web-app/en/admin/business_unit_new.jsp で CreateOrganizationFormHandler が使用されています。

新規ビジネス・ユニットのプロファイルに設定されたプロパティは、次のとおりです。

プロパティ	摘要
UserId	現在のユーザーの ID。このユーザーには、新しく作成される組織に対する管理者ロールが割り当てられます。
organizationName	作成されるビジネス・ユニットの名前。
parentOrganizationId	新しく作成されるビジネス・ユニットの親組織の ID。

これは、business_unit_new.jsp で使用されたフォームです。このページにアクセスする管理者に対して CreateOrganizationFormHandler.userId を設定する、非表示のフォーム・フィールドが作成されています。また、組織の名前、CreateOrganizationFormHandler.organizationName が設定されています。

```
<dsp:form action="business_units.jsp" method="post">
<dsp:input bean="CreateOrganizationFormHandler.userId"
   beanvalue="Profile.id" type="hidden"/>
<table border=0 cellpadding=4 cellspacing=0>
   <tr valign=top>
      <td align=right><span class=smallb>Name</span></td>
      <td><dsp:input bean="CreateOrganizationFormHandler.organizationName"
         size="30" type="text"/></td>
   </tr>
```

また、TargetPrincipals を使用して、ユーザーに組織のリストを表示します。ユーザーは新規ビジネス・ユニットの親をこのリストから選択して、parentOrganizationId を設定します。

```
   <tr valign=top>
      <td align=right><span class=smallb>Parent organization</td>
      <td>
      <dsp:select bean=
         "CreateOrganizationFormHandler.parentOrganizationId">
      <!--By default set the parent organization to current user's
          organization--!>
      <dsp:getvalueof id="parentId" idtype="java.lang.String"
         bean="Profile.parentOrganization.repositoryId">
      <dsp:option selected="<%=true%>" value="<%=parentId%>"/>Select
         Parent Organization
      </dsp:getvalueof>
      <!--Display all organizations available to this user--!>
      <dsp:droplet name="TargetPrincipals">
         <dsp:param bean="Profile.id" name="userId"/>
         <dsp:param name="roleName" value="admin"/>
         <dsp:oparam name="output">
         <dsp:droplet name="ForEach">
            <dsp:param name="array" param="principals"/>
            <dsp:param name="sortProperties" value="+name"/>
            <dsp:oparam name="output">
            <dsp:getvalueof id="parentId" idtype="java.lang.String"
               param="element.repositoryItem.repositoryId">
            <dsp:option value="<%=parentId%>"/>
            <dsp:valueof param="element.repositoryItem.name"/>
            </dsp:getvalueof>
            </dsp:oparam>
         </dsp:droplet>
         </dsp:oparam>
      </dsp:droplet>
      </dsp:select>
```

```
      </td>
   </tr>
```

さらに、「**Save**」ボタンと「**Cancel**」ボタンが付加されています。

```
   <tr valign=top>
      <td></td>
      <td><br>
      <dsp:input bean=
         "CreateOrganizationFormHandler.createOrganizationSuccessURL"
         type="hidden" value="business_units.jsp"/>

      <dsp:input bean=
         "CreateOrganizationFormHandler.createOrganizationErrorURL"
            type="hidden" value="business_unit_new.jsp"/>
      <dsp:input bean="CreateOrganizationFormHandler.createOrganization"
         type="submit" value=" Save "/>  
      <input type="submit" value=" Cancel "></td>

   </tr>

   </table>
</dsp:form>
```

組織ロールの設定

CreateOrganizationFormHandler の次のプロパティを使用して、新規ビジネス・ユニット内にロールが設定されます。

プロパティ	摘要
createRelativeRoles	新規組織に対して関連ロールが作成されるように、このプロパティはデフォルトで true に設定されます。
assignRelativeRoles	assignableFunctionNames で指定される関連ロールのリストがユーザーに割り当てられるように、このプロパティはデフォルトで true に設定されます。
functionNames	新規ビジネス・ユニットの関連ロールの名前を設定します。 たとえば、Motorprise の場合、これらのロールは admin、approver および buyer です。
assignableFunctionNames	新規ビジネス・ユニットを作成しているユーザーの関連ロールを指定します。Motorprise の場合、ユーザーにはそのビジネス・ユニットの admin ロールが割り当てられます。
userId	assignableFunctionNames でリストされる関連ロールが割り当てられるユーザーの ID です。

CreateOrganizationFormHandler の functionNames プロパティを使用して、新規組織のロールの名前を設定します。Motorprise の場合、これらのロールは admin、approver および buyer です。つまり、会社の管理者が Motorprise で新規ビジネス・ユニットを作成すると、次の名前の 3 つの関連ロールが作成されます。

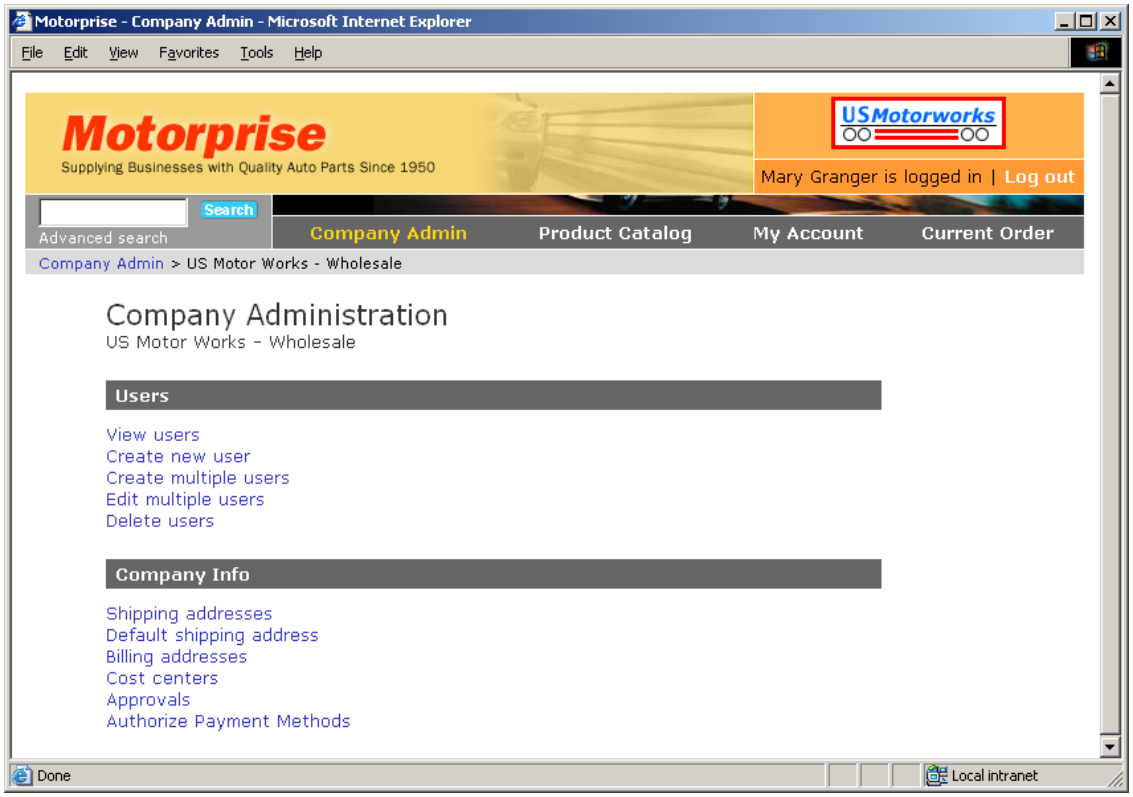
- **Buyer:** アプリケーションを使用して購買する、購買組織のメンバー
- **Approver:** 別のユーザーの購買を承認できる、購買組織のメンバー
- **Admin:** ストアの別のユーザーを作成および変更できる、購買組織のメンバー

これらのロールは、Motorprise のビジネス・モデルに適合します。業務にあわせて異なるロールを作成することもできます。詳細は、CreateOrganizationFormHandler に関する項を参照してください。

assignableFunctionNames プロパティは、userId プロパティで指定されるユーザーに割り当てるロールのリストです。これらの機能を使用して組織から関連ロールが抽出され、ユーザーに割り当てられます。この例では、ユーザーに admin のロールが割り当てられます。

ビジネス・ユニットの選択

USMW の Mary Granger などの管理者がログインし、「Company Admin」ページを選択すると、business_units.jsp に移動し、管理が許可されているすべてのビジネス・ユニットが表示されます。リスト内の各ビジネス・ユニットはリンクになっており、どのリンクをクリックしても、そのビジネス・ユニットの company_admin.jsp に移動します。そのページには、様々な管理オプションのリストが表示されます。



USMW - Wholesale ビジネス・ユニットの「Company Administration」ページ

この後に実行する個別の処理は、そのビジネス・ユニットに関係します。

ユーザーが business_units.jsp で選択した組織を判断するために、company_admin.jsp で atg/userdirectory/droplet/TargetPrincipals コンポーネントが使用されています。ユーザーがビジネス・ユニットのリンクを選択すると、その組織の ID がパラメータとして渡されます。TargetPrincipals は、ユーザーに管理者ロールが割り当てられている組織の配列を取得します。それぞれの ID を、パラメータとして渡された組織の ID と比較します。完全一致が見つかると、その組織が Profile.currentOrganization プロパティに割り当てられます。

company_admin.jsp からのこのスニペットは、TargetPrincipals の使用方法を示しています。

```
<%-- Set the Profile.currentOrganization property to the organization
which was selected in the business_units.jsp page --%>

<dsp:droplet name="IsEmpty">
  <dsp:param name="value" param="organizationId"/>
  <dsp:oparam name="false">
    <dsp:droplet name="TargetPrincipals">
      <dsp:param bean="Profile.id" name="userId"/>
      <dsp:param name="roleName" value="admin"/>
      <dsp:oparam name="output">
        <dsp:droplet name="ForEach">
        <dsp:param name="array" param="principals"/>
        <dsp:oparam name="output">
          <dsp:droplet name="Compare">
            <dsp:param name="obj1" param="organizationId"/>
              <dsp:param name="obj2" param=
               "element.repositoryItem.repositoryid"/>
            <dsp:oparam name="equal">
            <dsp:setvalue bean="Profile.currentOrganization"
              paramvalue="element.repositoryItem"/>

            </dsp:oparam>
          </dsp:droplet>

          </dsp:oparam>
        </dsp:droplet>  <%-- End of ForEach --%>
      </dsp:oparam>
    </dsp:droplet>  <%-- End of TargetPricipals --%>
  </dsp:oparam>
</dsp:droplet> <%-- End of IsEmpty --%>
```

複数のユーザーの登録

管理者は、複数のユーザーを共通の組織情報で同時に登録できる必要があります。たとえば、全員が「US Motor Works – Wholesale」グループのバイヤーである 5 人のユーザーを作成する場合があります。

これらのユーザーはすべて、出荷先所在地、請求先所在地、コスト・センターなど、親組織の様々なプロパティを継承します。

プロパティの継承

ユーザーを作成する場合、デフォルトで、選択されているビジネス・ユニットが親組織として設定されます。ユーザーはサブ組織と同様、親組織のプロパティを継承できます。ユーザー・プロファイル内の一部のプロパティは、ユーザー・レベルまたはサブ組織レベルで設定されていない場合、デフォルトで、この親組織から継承されます。出荷先所在地、請求先所在地、クレジット・カード、コスト・センター、購入限度などが、そのようなプロパティに含まれます。

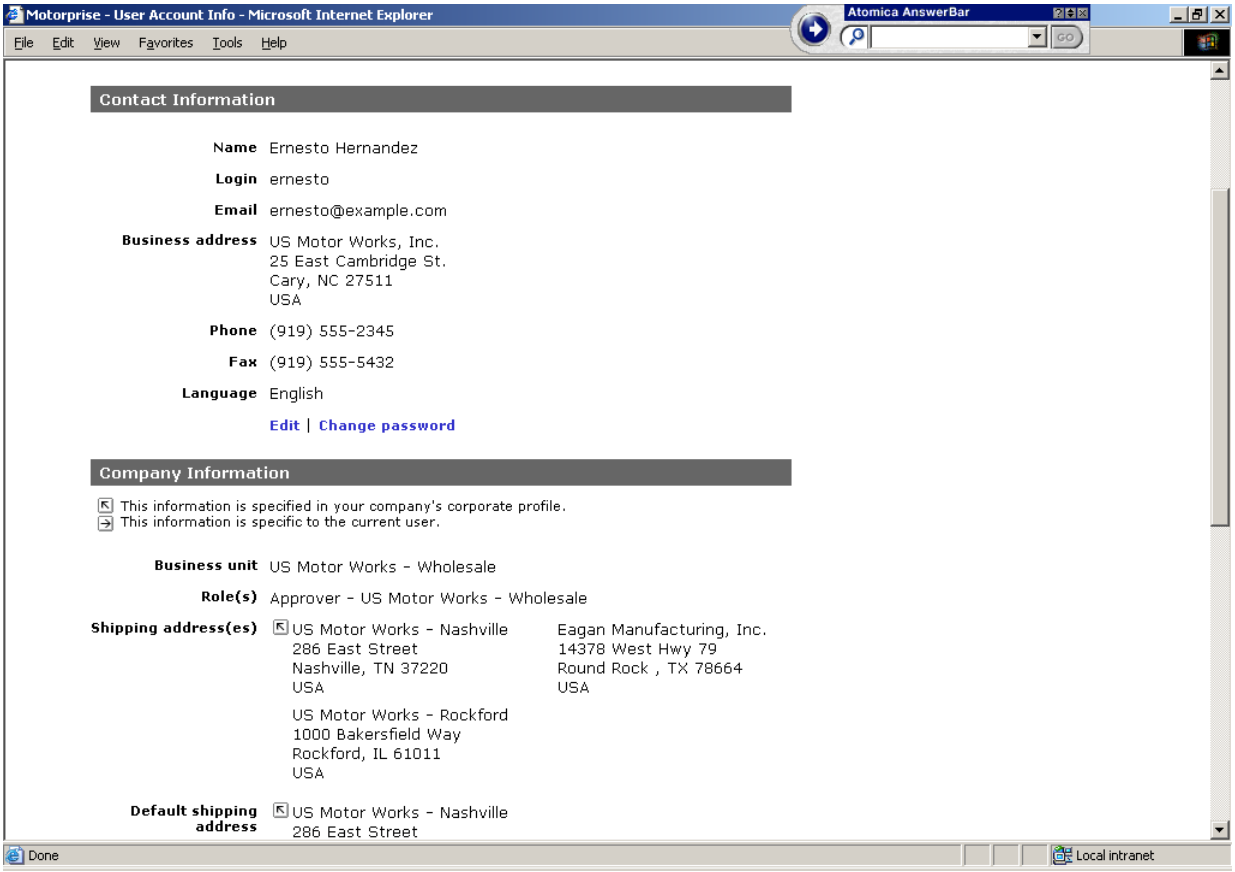
Motorprise サイトでユーザーのプロファイルまたは選択されているビジネス・ユニットの情報を表示すると、様々なフィールドの横に次の記号が表示されます。

この情報は会社の企業プロファイルに指定されています。

この場合、プロパティは現在のユーザーまたはサブ組織のレベルでは設定されておらず、親組織から継承されます。表示されているプロパティが現在のユーザーまたはサブ組織レベルの値を持たない場合が、これに該当します。

この情報は現在のユーザー固有のものです。

この場合、プロパティは現在のユーザーで設定されており、親組織から継承されていません。表示されているプロパティが現在のユーザーまたはサブ組織レベルの値を持つ場合が、これに該当します。ユーザーは後でプロファイルを変更でき、その結果、親組織レベルで設定されたプロパティを上書きできます。この操作が行われると、その変更を反映する記号が表示されます。



親組織から継承されたプロパティの Motorprise ユーザー・プロファイル内での表示

これらのユーザーが作成されると、各ユーザーがそれぞれのユーザー名で E メールを受信します。

新規ユーザーの作成

新規ユーザーを作成するときに入力する必要があるプロパティは、次のとおりです。

- FirstName
- LastName
- LoginId
- Password
- ConfirmPassword

次のプロパティはオプションです。

- Role
- Email
- Language（デフォルトは English）

Motorprise では、ユーザーの作成時に次のプロパティが親組織から継承されます。

- `invoiceRequestAuthorized`

- creditCardAuthorized
- giftCertificateAuthorized
- storeCreditAuthorized

グループ登録ページを作成するために、MultiUserAddFormHandler コンポーネントが使用されています。MultiUserAddFormHandler を使用すると、新規ユーザーの作成、組織へのユーザーの割当て、およびユーザーへのロールの割当てを行うことができます。

管理者がビジネス・ユニットを選択した後、「Create Multiple Users」リンクをクリックすると、複数のユーザーの追加を開始できます。最初に、管理者は作成するユーザーの数を入力する必要があります。このページ（create_multiple_users.jsp）の関連コードを、次に示します。

```
<dsp:setvalue bean="MultiUserAddFormHandler.clear" value=""/>

<table border=0 cellpadding=0 cellspacing=0 width=800>
    <tr valign=top>
    <td width=55><dsp:img src="../images/d.gif" hspace="27"/></td>

    <%--  main content area --%>
    <td valign="top" width=745>
    <dsp:form action="create_users2.jsp" method="post">
    <input type="hidden" name="b2bOp" value="add">
...

    <tr>
        <td>Create <dsp:input bean="MultiUserAddFormHandler.count"
          maxlength="1"
          size="1" type="text" value=""/> users.
        <p>
        <input type="submit" value="Create users">
        </td>
      </tr>
```

前述のコードに示すように、作成するユーザー数を管理者が入力し、それを使用して MultiUserAddFormHandler.count が初期化されます。この変数は次のページでも使用され、この数のユーザーを入力するために、その回数をループして入力フィールドを表示します。

管理者は、前述のページ（create_multiple_users.jsp）を発行すると create_users2.jsp に移動し、作成するすべてのユーザーの情報を入力します。MultiUserAddFormHandler.count 変数に基づいてユーザー配列を作成するために、MultiUserAddFormHandler が使用されています。次のコードは、MultiUserAddFormHandler.count を通じてループし、MultiUserAddFormHandler.users 配列内の各ユーザーにアクセスして、各ユーザーのプロパティを設定するための入力フィールドを表示する方法を示しています。

```
<dsp:form action="create_multiple_users.jsp" method="post">

  <dsp:input bean="MultiUserAddFormHandler.confirmPassword" type="hidden"
    value="true"/>
  <dsp:input bean="MultiUserAddFormHandler.createErrorURL" type="hidden"
    value="create_users2.jsp"/>
```

```
<dsp:input bean="MultiUserAddFormHandler.createSuccessURL" type="hidden"
  value="create_users3.jsp"/>
<dsp:input bean="MultiUserAddFormHandler.organizationId" beanvalue=
  "Profile.currentOrganization.repositoryid" type="hidden"/>
<dsp:input bean="MultiUserAddFormHandler.value.member" type="hidden"
  value="true"/>

<table border=0 cellpadding=4 width=100%>
  <tr><td><dsp:img src="../images/d.gif" vspace="0"/></td></tr>
  <tr valign=top>
    <td colspan=2><span class=big>Company Administration</span><br><span
      class=little></span></td>
  </tr>
  <tr><td><dsp:img src="../images/d.gif" vspace="0"/></td></tr>

  <tr valign=top>
    <td colspan=2>
    <table width=100% cellpadding=3 cellspacing=0 border=0>
      <tr><td class=box-top> Create New Users</td></tr></table>
    </td>
  </tr>
  <tr><td><dsp:img src="../images/d.gif" vspace="0"/></td></tr>

   <!--Display all the users--!>
   <dsp:droplet name="/atg/dynamo/droplet/For">
    <dsp:param bean="MultiUserAddFormHandler.count" name="howMany"/>
     <dsp:oparam name="output">
       <tr>
         <td align=right><span class=smallb>User <dsp:valueof
           param="count"/>
           </span></td>
       </tr>
       <tr>
         <td align=right><span class=smallb>Name</span></td>
         <td width=75%><dsp:input bean=
           "MultiUserAddFormHandler.users[param:index].value.firstName"
           size="15" type="text" required="<%=true%>"/>
          <dsp:input bean=
           "MultiUserAddFormHandler.users[param:index].value.middleName"
              size="4" type="text"/>
           <dsp:input bean=
             "MultiUserAddFormHandler.users[param:index].value.lastName"
               size="15" type="text" required="<%=true%>"/></td>
       </tr>
       <tr>
         <td align=right><span class=smallb>Login</span></td>
         <td><dsp:input bean=
           "MultiUserAddFormHandler.users[param:index].value.login"
            size="30"
```

```
      type="text"/></td>
</tr>
<tr>
  <td align=right><span class=smallb>Password</span></td>
  <td><dsp:input bean=
    "MultiUserAddFormHandler.users[param:index].value.Password"
     size="30"
      type="password" value=""/></td>
</tr>
<tr>
  <td align=right><span class=smallb>Confirm</span></td>
  <td>
    <dsp:input bean=
      "MultiUserAddFormHandler.users[param:index].
       value.CONFIRMPASSWORD"
       size="30" type="password" value=""/>
  </td>
</tr>
<tr>
  <td align=right><span class=smallb>Email</span></td>
  <td><dsp:input bean=
   "MultiUserAddFormHandler.users[param:index].value.email"
     size="30"
      type="text"/></td>
</tr>
<tr valign=top>
  <td align=right><span class=smallb>Role</span></td>
  <td>

       <%/* Check if the roleIds already exist.  This property will not
        be set when the page is displayed for the first time.  In this
        case, display all the roles as unchecked boxes.  If this property
        is set, that means this page has been displayed already and that
        an error has occured.  In this case, each role in the
        currentOrganization is traversed to see if any of their ids exist
        in the roleIds property and if so, display that role as a checked
        box. */ %>

       <dsp:droplet name="IsEmpty">
       <dsp:param bean=
         "MultiUserAddFormHandler.users[param:index].roleIds"
         name="value"/>
       <dsp:oparam name="true">

       <%/* List organization roles, allowing admin to check off
            roles for each new user */%>
          <dsp:droplet name="ForEach">
          <dsp:param bean=
           "Profile.currentOrganization.relativeRoles"
```

```
      name="array"/>
    <dsp:param name="elementName" value="role"/>
    <dsp:param name="indexName" value="roleIndex"/>
    <dsp:oparam name="output">

       <dsp:input bean=
       "MultiUserAddFormHandler.users[param:index].roleIds"
         paramvalue="role.repositoryId" type="checkbox" />
         <dsp:valueof param="role.name">No name
       </dsp:valueof>
    </BR>
    </dsp:oparam>
    </dsp:droplet>
</dsp:oparam>

<dsp:oparam name="false">

    <dsp:droplet name="ForEach">
    <dsp:param bean="Profile.currentOrganization.
     relativeRoles"
      name="array"/>
    <dsp:param name="elementName" value="role"/>
    <dsp:param name="indexName" value="roleIndex"/>
    <dsp:oparam name="output">

       <dsp:droplet name="ArrayIncludesValue">
       <dsp:param bean=
       "MultiUserAddFormHandler.users[param:index].roleIds"
         name="array"/>
       <dsp:param name="value" param="role.repositoryId"/>
       <dsp:oparam name="true">
         <dsp:input bean=
       "MultiUserAddFormHandler.users[param:index].roleIds"
           paramvalue="role.repositoryId" type="checkbox"
           checked="<%=true%>"/>  <dsp:valueof param=
           "role.name">No
           name</dsp:valueof>
       </dsp:oparam>
       <dsp:oparam name="false">
            <dsp:input bean=
       "MultiUserAddFormHandler.users[param:index].roleIds"
          paramvalue="role.repositoryId" type="checkbox" />
       <dsp:valueof param="role.name">No name</dsp:valueof>
       </dsp:oparam>
       </dsp:droplet>
     </br>
     </dsp:oparam>
     </dsp:droplet>
```

```
                    </dsp:oparam>
                    </dsp:droplet>
          </td>
        </tr>
        <tr>
          <td align=right><span class=smallb>Language</span></td>
          <td><dsp:select bean=
            "MultiUserAddFormHandler.users[param:index].value.locale">
                <dsp:option value="en_US"/>English
                <dsp:option value="de_DE"/> German
              </dsp:select></td>
        </tr>

        <tr>
          <td colspan=2><hr size=1 color="#666666"></td>
        </tr>
        <tr><td><dsp:img src="../images/d.gif" vspace="6"/></td></tr>
      </dsp:oparam>
    </dsp:droplet>
    <tr>
      <td></td>
      <td><b><dsp:input bean="MultiUserAddFormHandler.create"
        type="submit"
        value=" Save "/>  
               <input type="submit" value=" Cancel " ></td>
    </tr>
    <%--  End of add new user action --%>

</table>

</dsp:form>
```

単一値プロパティの設定

前述のコードは、firstNameなど、ユーザー・プロファイル内で単一の値を持つプロパティ値を設定する方法を示しています。MultiUserAddFormHandler.users[param:index].value.propertyNameの形式が使用されています。ここで、indexは、作成されるユーザーの配列内の現在のユーザーを指定する数値です。たとえば、MultiUserAddFormHandler.users[0].value.firstNameは、先頭のユーザーのfirstNameを設定します。

たとえば、MultiUserAddFormHandler.countプロパティが10に設定されている場合、users[ ]配列には10個のエントリがあり、同じフォームの発行時に10ユーザーのプロパティを設定できます。

create操作はMultiUserAddFormHandler.createに発行するsubmitボタンによって起動されます。管理者はグループ登録ページを使用して単一のユーザーを登録することもできます。

配列、リスト、または他の項目のマップであるプロパティの設定

プロパティが配列、リスト、または他の項目のマップの場合、Webページからリポジトリ項目を割り当てることができないため、別の方法でそれを設定する必要があります。配列、リスト、または他の項目のマップである

プロパティを設定するために、既存の項目を参照する ID が使用されています。repositoryIds という、プロパティの特別なサブプロパティが設定されています。

たとえば、user の roles プロパティはマップであり、次のように設定されています。

```
<dsp:input type="checkbox" value="role0001"
   bean="MultiUserAddFormHandler.users[0].value.roles.repositoryIds"/>
<dsp:input type="checkbox" value="role0002"
   bean="MultiUserAddFormHandler.users[0].value.roles.repositoryIds"/>
```

前述のコードでは、タイプが role、ID が role001 および role002 のリポジトリ項目が、リポジトリ内に存在します。これらのロール ID は、user の roles プロパティの repositoryIds サブプロパティとして設定されています。これはマップであり、フォーム・ハンドラが対応するリポジトリ項目を自動的に取得し、マップに追加します。

複数のユーザー・プロパティの編集

「Edit Multiple Users」ページを使用すると、複数のバイヤーのプロパティを更新できます。これは、複数のユーザーのプロファイル・プロパティを同じ値に設定する場合に便利です。たとえば、20 ユーザーの購入限度を$5000 に設定するものとします。「Edit Multiple Users」機能を使用すると、1 回のフォーム発行でこれを行うことができます。

Motorprise におけるグループ更新のために、atg/userprofiling/MultiUserUpdateFormHandler コンポーネントが使用されています。このコンポーネントには、プロファイルが更新されるユーザーのリポジトリ ID が必要です。これらは repositoryids プロパティで設定されます。

MultiUserUpdateFormHandler の詳細は、『ATG Web Commerce Platform API Reference』を参照してください。

プロファイル・プロパティは MultiUserUpdateFormHandler.value ディクショナリを使用して更新されます。このディクショナリ内の値は、そのユーザーID が repositoryIds プロパティに含まれているすべてのユーザーに適用されます。たとえば、そのリポジトリ ID が MultiUserUpdateFormHandler.repositoryIds に設定されているすべてのユーザーの orderPriceLimit を更新するには、次の形式を使用できます。
MultiUserUpdateFormHandler.value.orderPriceLimit

配列、集合、リスト、または他の項目のマップの場合は、ID を使用して既存の項目を参照して設定します。repositoryIds という、プロパティの特別なサブプロパティを設定してこれを行います。たとえば、ユーザーのロールを設定するには、これがマップの場合は次のような input タグを使用します。

```
<dsp:input type="checkbox" bean="MultiProfileUpdateFormHandler.
  value.roles.repositoryIds" value="r001"/>
```

更新処理は、MultiProfileUpdateFormHandler.update を実行する発行ボタンを使用して呼び出されます。

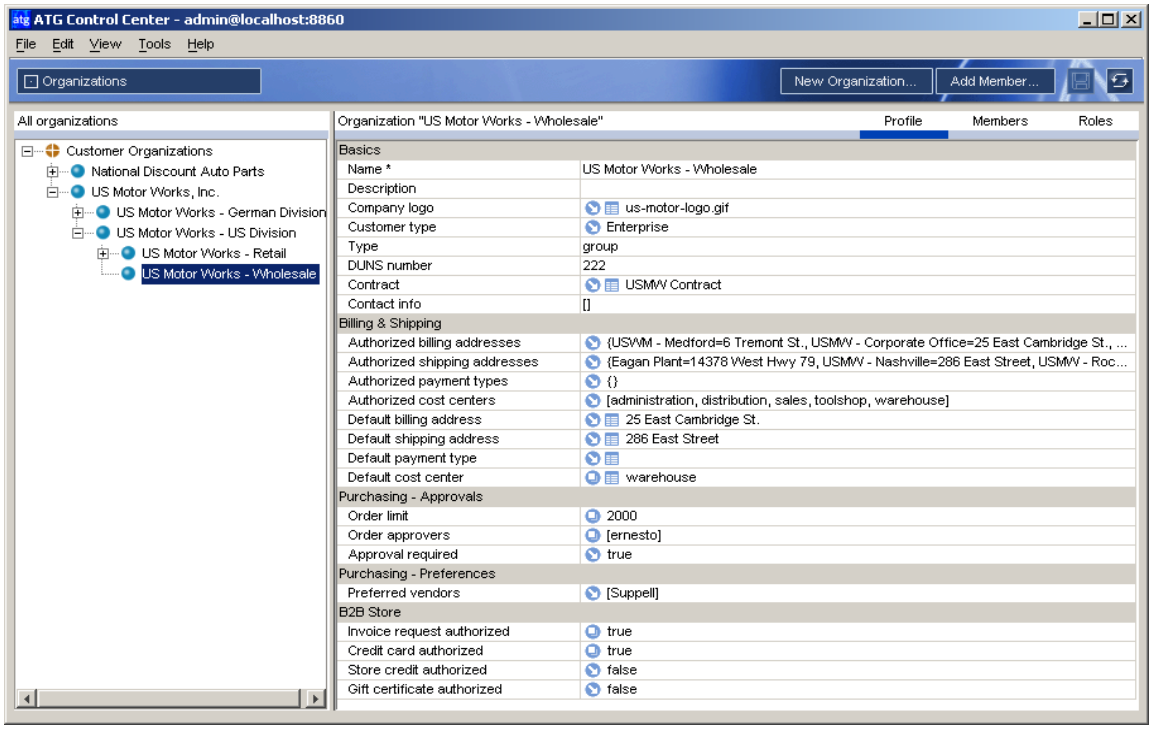
次のコードは、ユーザーID が Id001 および Id002 であるユーザーの、OrderPriceLimit プロパティとロールを設定する方法を示しています。

```
<%@ taglib uri="http://www.atg.com/dsp.tld" prefix="dsp" %>
<dsp:page>
```

```
<dsp:form action="xxx" method="post">
   <dsp:input bean="MultiUserUpdateFormHandler.repositoryIds"
      type="checkbox" value="Id001"/>
   <dsp:input bean="MultiUserUpdateFormHandler.repositoryIds"
      type="checkbox" value="Id002"/>
   <dsp:input bean="MultiUserUpdateFormHandler.value.OrderPriceLimit" type="text"
      value=""/>
   <dsp:input bean="MultiUserUpdateFormHandler.value.roles.repositoryIds"
      name="Admin" type="checkbox" value="R001"/>
   <dsp:input bean="MultiUserUpdateFormHandler.value.roles.repositoryIds"
      name="Buyer" type="checkbox" value="R002"/>
   <dsp:input bean="MultiUserUpdateFormHandler.update" type="submit"
      value="submit"/>
</dsp:form>
</dsp:page>
```

ユーザーの削除

Motorprise ユーザーは承認者や管理者のロールを持つことができるため、親組織のプロファイル内で参照される可能性があります。たとえば、Ernesto Hernandez は、USMW - Wholesale のプロファイル内で承認者として表示されます。



Ernesto Hernandez は USMW - Wholesale プロファイル内でオーダー承認者として表示される

ユーザーを削除するときは、ユーザーへのこのような参照もすべて削除される必要があります。ユーザーを削除するときにこれらの参照が削除されないと、参照整合性制約違反が生じます。

この状態を回避するために、atg.projects.b2bstore.userprofiling.B2BDeleteFormHandler という新しいクラスが作成されています。これは、MultiProfileUpdateFormHandler を拡張し、ユーザー・プロファイルが削除されるときの動作を上書きします。ベース・クラスに制御を移してユーザー項目を削除する前に、B2BDeleteFormHandler 内でユーザー項目への参照を削除します。B2BDeleteFormHandler の使用方法と MultiProfileUpdateFormHandler の使用方法に違いはありません。Motorprise では、「Company Administration」ページの「Delete Users」リンク（users_delete.jsp）を使用してユーザーを削除できます。すべてのユーザーとチェック・ボックスのリスト、および選択されたユーザーを削除する「Delete」ボタンが表示されます。

次のコードは、users_delete.jsp からのコード・スニペットです。

最初に、組織のリポジトリ ID が atg/userdirectory/droplet/UserList コンポーネントに渡されて、現在の組織のユーザーのリストが取得されます。ページにアクセスしている管理者の ID が excludeId に渡されるため、返されるユーザーのリストから管理者は除外されます。このため、管理者のログイン中にそのアカウントが削除されることはありません。

```
<dsp:droplet name="UserList">
   <dsp:param bean="Profile.currentOrganization.repositoryid"
      name="organizationId"/>
<dsp:param bean="Profile.id" name="excludeId"/>
   <dsp:oparam name="output">
```

次に、ユーザー・リポジトリ項目の名前をフォーム・ハンドラに設定して削除します。

```
<dsp:form action="users_delete_confirm.jsp" method="post">
   <dsp:input bean="DeleteProfileFormHandler.itemDescriptorName"
      type="hidden" value="user"/>
```

毎回 10 項目を取得し、各ユーザーを表示するために、Range ドロップレットが使用されています。

```
<dsp:droplet name="Range">
   <dsp:param name="array" param="users"/>
   <dsp:param name="start" param="startIndex"/>
   <dsp:param name="howMany" value="10"/>
   <dsp:param name="sortProperties" value="+name"/>
```

次に、チェック・ボックスを表示します。管理者がページ上のこのボックスを選択すると、このユーザー項目がリポジトリから削除されるように、対応するユーザーのリポジトリ ID が DeleteProfileFormHandler.repositoryIds に追加されます。

```
<dsp:oparam name="output">
<dsp:input bean="DeleteProfileFormHandler.repositoryIds"
   paramvalue="element.repositoryItem.id" type="checkbox"/>
   <dsp:valueof param="element.repositoryItem.firstName"/> <dsp:valueof
      param="element.repositoryItem.lastName"/><BR>
</dsp:oparam>

<dsp:oparam name="outputEnd">
   <dsp:droplet name="IsEmpty">
```

```
      <dsp:param name="value" param="nextHowMany"/>
      <dsp:oparam name="false"><BR>
        <dsp:a href="users_delete.jsp">  Next 10 <dsp:param name="startIndex"
          param="nextStart"/></dsp:a>
      </dsp:oparam>
    </dsp:droplet>
</dsp:oparam>
```

前述のコードで表示されたユーザーを選択し、「Delete」ボタンをクリックすると、削除の確認画面（users_delete_confirm.jsp）にリダイレクトされます。この画面には確認のためにユーザーと「Delete」ボタンが表示されます。

前の画面で削除が選択されたすべてのユーザーのリポジトリ ID を取得し、ProfileLookUp コンポーネントを使用してプロファイルをフェッチして、ユーザー情報を表示します。

```
<dsp:form action="company_admin.jsp" method="post">

   <dsp:droplet name="ForEach">
      <dsp:param bean="DeleteProfileFormHandler.repositoryIds" name="array"/>
      <dsp:param name="elementName" value="userId"/>
      <dsp:oparam name="output">
         <dsp:droplet name="ProfileLookUp">
            <dsp:param name="id" param="userId"/>
            <dsp:param name="elementName" value="user"/>
            <dsp:oparam name="output">
            <dsp:valueof param="user.firstName"/>
             
            <dsp:valueof param="user.lastName"/><br>
            </dsp:oparam>
         </dsp:droplet>
      </dsp:oparam>
   </dsp:droplet>

<br>
   <dsp:input bean="DeleteProfileFormHandler.delete" type="submit"
      value="Delete"/>  
   <dsp:input bean="DeleteProfileFormHandler.cancel" type="submit"
      value="Cancel"/>
</dsp:form>
```

B2BRepositoryFormHandler の使用

Oracle Commerce Platform に含まれている RepositoryFormHandler を使用すると、任意のリポジトリのリポジトリ項目を作成、更新および削除できます。Motorprise では、新規作成したリポジトリ項目を他の項目のプロパティに割り当てる必要があるため、このクラスが拡張され、atg/projects/b2bstore/repository/ に B2BRepositoryFormHandler が作成されています。B2BRepositoryFormHandler は、既存のリポジ

トリ項目と、このフォーム・ハンドラによって作成および削除されるリポジトリ項目との間の関係を自動的に管理する上で役立ちます。

（RepositoryFormHandler の詳細は、『ATG Web Commerce Page Developer's Guide』を参照してください）

B2BRepositoryFormHandler には、リポジトリ項目への参照の追加および削除という 2 つの重要な機能があります。

リポジトリ項目への参照の追加

B2BRepositoryFormHandler は、新規作成されたリポジトリ項目を既存のリポジトリ項目のプロパティに自動的に追加できます。たとえば、このフォーム・ハンドラを使用して、組織と出荷先所在地との間の関係を管理できます。Motorprise で新規の出荷先所在地を作成すると、B2BRepositoryFormHandler はそれを選択されている任意のビジネス・ユニットに割り当てることができます。

リポジトリ項目への参照の削除

B2BRepositoryFormHandler は、リポジトリからリポジトリ項目が削除されるときに、リポジトリ項目への参照を削除できます。deleteItem を上書きして、各リポジトリ項目を削除する前に、それぞれへの参照をチェックします。最後の参照がなくなった場合のみ、リポジトリから項目が実際に削除されます。B2B Commerce では、多くの場合、複数の組織が同じ項目を参照します。たとえば、USMW Retail 部と USMW Wholesale 部は、出荷先所在地を共有する可能性があります。Retail 部からこの出荷先所在地を削除しても、Wholesale 部が引き続き参照しているため、リポジトリからこれが削除されることはありません。ただし、すべての部署から所在地が削除された場合は、リポジトリから削除する*必要があります*。B2BRepositoryFormHandler は deleteItem メソッドにロジックを追加して、リポジトリ項目への参照がすべて削除されるまで、リポジトリ項目が削除されないようにしています。

「Company Admin」における B2BRepositoryFormHandler の使用

Motorprise の「Company Admin」の機能は、B2BRepositoryFormHandler を使用して提供されています。管理者がログインしてビジネス・ユニットを選択すると、そのビジネス・ユニット内で次のものを管理できます。

機能	摘要
出荷先所在地	新規作成。 既存の項目の編集または削除。
デフォルト出荷先所在地	既存の出荷先所在地のリストからのデフォルト出荷先所在地の選択。
請求先所在地	新規作成。 既存の項目の編集または削除。
デフォルト請求先所在地	既存の請求先所在地のリストからのデフォルト請求先所在地の選択。
クレジット・カード	新規作成。 既存の項目の編集または削除。
コスト・センター	新規作成。 既存の項目の編集または削除。

機能	摘要
承認	必須かどうかの指定。 購入限度の変更。
支払方法	請求書およびクレジット・カードの使用の許可または制限。

たとえば、shipping_address_create.jsp からの次のコードでは、新規出荷先所在地を作成し、管理者が所属する組織にその出荷先所在地を追加します。このコードについて順を追って説明します。

この例では、タイプ contactInfo の新規リポジトリ項目が作成され、タイプが organization でリポジトリ ID が Profile.currentOrganization.repositoryid であるリポジトリ項目の、shippingAddrs プロパティに割り当てられます。

address1 は contactInfo プロパティで、B2BRepositoryFormHandler の value ディクショナリを使用してアクセスできます。これは、requiredFields プロパティの一部でもあります。

updateItemDescriptorName プロパティを使用して、新規作成した出荷先所在地を追加している項目タイプの名前を設定します。この例では organization です。

```
<%--  main content area --%>
<td valign="top" width=745>
<dsp:include page="../common/FormError.jsp" flush="true"></dsp:include>
  <dsp:form action="shipping_edit.jsp" method="post">
    <dsp:input bean="B2BRepositoryFormHandler.itemDescriptorName" type="hidden"
      value="contactInfo"/>
    <dsp:input bean="B2BRepositoryFormHandler.updateItemDescriptorName"
      type="hidden" value="organization"/>
```

updateRepositoryId プロパティを使用して、新規所在地が追加される項目のリポジトリ ID を設定します。次の例では、これは親組織のリポジトリ ID の Profile.currentOrganization.repositoryid です。

updatePropertyName によって、新規作成された出荷先所在地が追加される親組織のプロパティが決まります。この例では、出荷先所在地のリストが含まれる、親組織のリスト・プロパティの shippingAddrs に設定されます。

```
    <dsp:input bean="B2BRepositoryFormHandler.updateRepositoryId"
      beanvalue="Profile.currentOrganization.repositoryid" type="hidden"/>
    <dsp:input bean="B2BRepositoryFormHandler.updatePropertyName" type="hidden"
      value="shippingAddrs"/>
```

フォーム内の 1 つ以上のフィールドが null 以外の値を持つ必要がある場合は、RequiredFields を設定すると、ユーザーがフォームを発行する前に値を入力することが常に要求されます。

shippingAddrs を使用するこの場合のように、変更されているプロパティがリストではなくマップの場合は、新規マップ・エントリに対してキーを提供する必要があります。これは updateKey プロパティで指定されます。

```
    <dsp:input bean="B2BRepositoryFormHandler.requireIdOnCreate" type="hidden"
      value="false"/>
```

```
  <dsp:input bean="B2BRepositoryFormHandler.requiredFields" name="hiddenFields"
    type="hidden" value="COMPANYNAME"/>
  <dsp:input bean="B2BRepositoryFormHandler.requiredFields" name="hiddenFields"
    type="hidden" value="ADDRESS1"/>
  <dsp:input bean="B2BRepositoryFormHandler.requiredFields" name="hiddenFields"
    type="hidden" value="CITY"/>
  <dsp:input bean="B2BRepositoryFormHandler.requiredFields" name="hiddenFields"
    type="hidden" value="POSTALCODE"/>
  <dsp:input bean="B2BRepositoryFormHandler.createErrorURL" type="hidden"
    value="shipping_address_create.jsp"/>
  <dsp:input bean="B2BRepositoryFormHandler.createSuccessURL" type="hidden"
    value="shipping_edit.jsp"/>

 <table border=0 cellpadding=4 width=80%>
   <tr>
     <td colspan=2><dsp:img src="../images/d.gif" vspace="0"/></td>
   </tr>
   <tr>
     <td colspan=2 valign="top"><span class=big>Company Administration</span>
       <br><span class=little><dsp:valueof
       bean="Profile.currentOrganization.name" /></span></td>
   </tr>
   <tr>
     <td colspan=2><dsp:img src="../images/d.gif" vspace="0"/></td>
   </tr>

    <tr>
      <td colspan=2 valign="top">
        <table width=100% cellpadding=3 cellspacing=0 border=0>
          <tr>
            <td class=box-top> Create Shipping Address</td>
          </tr>
        </table>
      </td>
    </tr>
<tr>
  <td colspan=2><span class=small>The nickname field is used to identify this
    address when you don't have access to the full address. It should be unique
    from all other shipping address nicknames. It can be the same as the company
    name.</span></td>
</tr>

    <tr>
      <td colspan=2><dsp:img src="../images/d.gif" vspace="0"/></td>
    </tr>
    <tr>
      <td align=right><span class=smallb>Nickname</span></td>
      <td width=75%><dsp:input bean="B2BRepositoryFormHandler.updateKey"
```

フォーム・ハンドラのスコープの変更

Motorprise では、ストアの「Company Admin」セクションにおける複数ページの登録プロセスに対処するために、`MultiUserAddFormHandler` コンポーネントと `MultiUserUpdateFormHandler` コンポーネントの `$scope` が `request` から `session` に上書きされています。

様々なページで入力される情報は、プロセス全体をとおして保持される必要があります。この情報はフォーム・ハンドラに格納されるため、それらは session-scoped になっています。

実際のコンポーネント内でのスコープの上書きが選択されたのは、Motorprise が実行されている唯一のアプリケーションであるためです。標準の構成をコピーし、自分のバージョン内でスコープをセッションに変更することによって、これらのコンポーネントの独自のインスタンスを構成することもできます。

元のコンポーネント内でスコープを変更すると、Oracle Commerce Platform の新バージョンにアップグレードした場合、またはフォーム・ハンドラに新規プロパティを追加するパッチやデフォルト設定を変更するパッチをインストールした場合に、特別な処理を実行することなく変更を反映できます。

しかし、同じサーバー上で別のアプリケーションが実行されている場合、元のコンポーネント内でスコープを上書きすると、request-scoped のフォーム・ハンドラが予想される場所で session-scoped のフォーム・ハンドラを使用することになります。

これらのコンポーネントの独自のインスタンスを構成するには、標準の構成をコピーし、自分のバージョン内でスコープをセッションに変更する必要があります。

これを行うには、各デフォルト・コンポーネントの構成全体を、`/atg/projects/B2BStore/userprofiling/MultiProfileFormHandler` など、アプリケーション固有のパスにコピーし、そのコンポーネントのスコープを変更し、JSP 内でそのコンポーネント・パスを使用します。

`^=`を使用すると、すべてのプロパティ値が元のコンポーネントを参照するように設定できます。次に例を示します。

```
trimProperties^=/atg/userprofiling/MultiProfileAddFormHandler.trimProperties
```

この方法の場合、正確な値を知る必要はなく、コンポーネントの構成にすべてを確実にコピーすることのみ必要です。`^=`を使用すると、あるオブジェクトから別のオブジェクトへプロパティ値がコピーされます。このコマンドの詳細は、『ATG Web Commerce Platform Programming Guide』を参照してください。

ただし、独自のオブジェクトを作成する場合は、新バージョンやパッチをインストールする場合および更新する場合に、構成を忘れずにチェックして、元のコンポーネントに追加または削除されている可能性があるプロパティを把握する必要があります。

6　自分のアカウント

Motorprise サイトの「My Account」セクションでは、ユーザーが各自のオープン・オーダーおよび履行済オーダーをレビューできます。また、購入リストを作成することや、プロファイル情報を編集することもできます。さらに、承認者である Motorprise ユーザーは、「My Account」セクションで承認を管理できます。

この章では、「My Account」セクションをどのように作成したかについて説明します。この章は次の項から構成されています。

オーダーのレビュー
オープン、履行済および否認済のオーダーの表示方法について説明します。

購入リストの使用
購入リスト機能について説明します。

オーダーのスケジュール
ユーザーがオーダーをスケジュールする方法について説明します。

オーダーの保存
ユーザーがオーダーを保存する方法について説明します。

オーダーの承認
承認プロセスについて説明します。

オーダーのレビュー

バイヤーが Motorprise サイトにログインすると、オープン、履行済および否認済のオーダーがホーム・ページに表示されます。

バイヤーがホーム・ページにログインすると、オープン、履行済および否認済のオーダーが表示される。

Motorprise バイヤーは、ストアの「My Account」セクションでオーダーを表示することもできます。各オーダーの番号、日付およびステータスが表示されます。

Motorprise バイヤーの「My Account」セクションにおけるオープン・オーダー

バイヤーはオーダー番号をクリックして、そのオーダーの詳細を表示できます。

オーダーのリストを表示するために、`MotorpriseJSP/j22ee-apps/motorprise/web-app/en/users/orders_open.jsp` の次の JSP コード・フラグメントが使用されています。

```
<dsp:getvalueof id="profileId" idtype="java.lang.String" bean="Profile.id">
  <%-- profileId:  <%=profileId%> --%>
  <dsp:droplet name="OrderLookup">
    <dsp:param value="<%=profileId%>" name="userId"/>
    <dsp:param name="state" value="open"/>
    <dsp:param name="startIndex" param="startIndex"/>
    <dsp:oparam name="output">
      <dsp:droplet name="ForEach">
        <dsp:param name="array" param="result"/>
        <dsp:oparam name="outputStart">
          <table border=0 width=95%>
            <tr valign=top>
              <td width=33%><span class="small"><b>Order #</b></span></td>
              <td width=33%><span class="small"><b>Date ordered</b></span></td>
              <td width=33%><span class="small"><b>Status</b></span></td>
            </tr>
            <tr>
              <td colspan=3><hr size=1 color="#666666"></td>
            </tr>

        </dsp:oparam>
```

```
<dsp:oparam name="output">
  <tr>
    <dsp:droplet name="Switch">
    <dsp:param name="value" param="element.originOfOrder"/>
    <dsp:oparam name="default">
      <td><dsp:a href="order.jsp">
      <dsp:param name="orderId" param="element.id"/>
      <dsp:valueof param="element.id"/><dsp:param name="orderState"
        param="state"/>
      </dsp:a></td>
    </dsp:oparam>
    <dsp:oparam name="scheduledOrder">
      <td><dsp:a href="order.jsp">
      <dsp:param name="orderId" param="element.id"/>
      <dsp:valueof param="element.id"/>
      </dsp:a>(ScheduledOrder)</td>
    </dsp:oparam>
    </dsp:droplet>
    <td><dsp:valueof date="MMMMM d, yyyy"
    param="element.submittedDate"/></td>
    <td><dsp:valueof param="element.stateAsString"/></td>
  </tr>

</dsp:oparam>
<dsp:oparam name="outputEnd">
  <tr></tr>
  <tr>
    <td colspan=3>
        <hr size=1 color="#666666">
        Now viewing orders
        <dsp:valueof param="startRange"/> -
        <dsp:valueof param="endRange"/> out of
        <dsp:valueof param="total_count"/>
    </td>
    <td>
        <dsp:droplet name="IsEmpty">
          <dsp:param name="value" param="previousIndex"/>
          <dsp:oparam name="false">
            <dsp:a href="orders_open.jsp">
            <dsp:param name="startIndex" param="previousIndex"/>
            Previous</dsp:a>
          </dsp:oparam>
        </dsp:droplet>

        <dsp:droplet name="IsEmpty">
          <dsp:param name="value" param="nextIndex"/>
          <dsp:oparam name="false">
            <dsp:a href="orders_open.jsp">
            <dsp:param name="startIndex" param="nextIndex"/>
```

```
                    Next</dsp:a>
                  </dsp:oparam>
                </dsp:droplet>
              </td>
            </tr>
          </dsp:oparam>


      </dsp:droplet>
    </dsp:oparam>
    <dsp:oparam name="empty">
      <%/* no open orders for user */%>
        You have no open orders.
    </dsp:oparam>
    <dsp:oparam name="error">
      <dsp:valueof param="errorMsg"/>
    </dsp:oparam>
  </dsp:droplet>
</dsp:getvalueof>
```

ユーザーのオーダーをフェッチするためには、OrderLookup コンポーネントが使用されています。これは各オーダーの userId、startIndex および state を受け取ってフェッチし、numOrders パラメータが与えられていない場合は defaultNumOrders が含まれる結果配列を出力します。

ユーザーのオーダーが数百にのぼる可能性があるため、それらを簡単に参照できる必要があります。そこで、オーダーを 10 件のグループ単位で表示し、前と次のグループに移動するリンクを一緒に表示することになりました。オーダーにおけるこの種のページ・ナビゲーションを実現するために、OrderLookup のパラメータが使用されています。

OrderLookup によって返されるオーダー数は、パラメータ numOrders で決まります。この値が設定されていない場合は、プロパティ defaultNumOrders によって返されるオーダー数が決まります。さらに、startIndex パラメータを使用して、返されるオーダーの先頭のインデックスを指定します。OrderLookup を使用して、パラメータ previousIndex および nextIndex を指定します。

また、OrderLookup を出力の他のパラメータに対しても使用して、一度に特定の数のオーダーのみを表示します。これらのパラメータに基づいて、前述のコードに示すように、前のページおよび次のページ用のリンクを提供して一連のオーダーを表示します。

すべてのオーダーを 1 ページに表示する場合は、OrderLookup の defaultNumOrders プロパティを–1 に設定すると、出力パラメータの結果配列に、特定のユーザーの特定の状態のすべてのオーダーが格納されます。

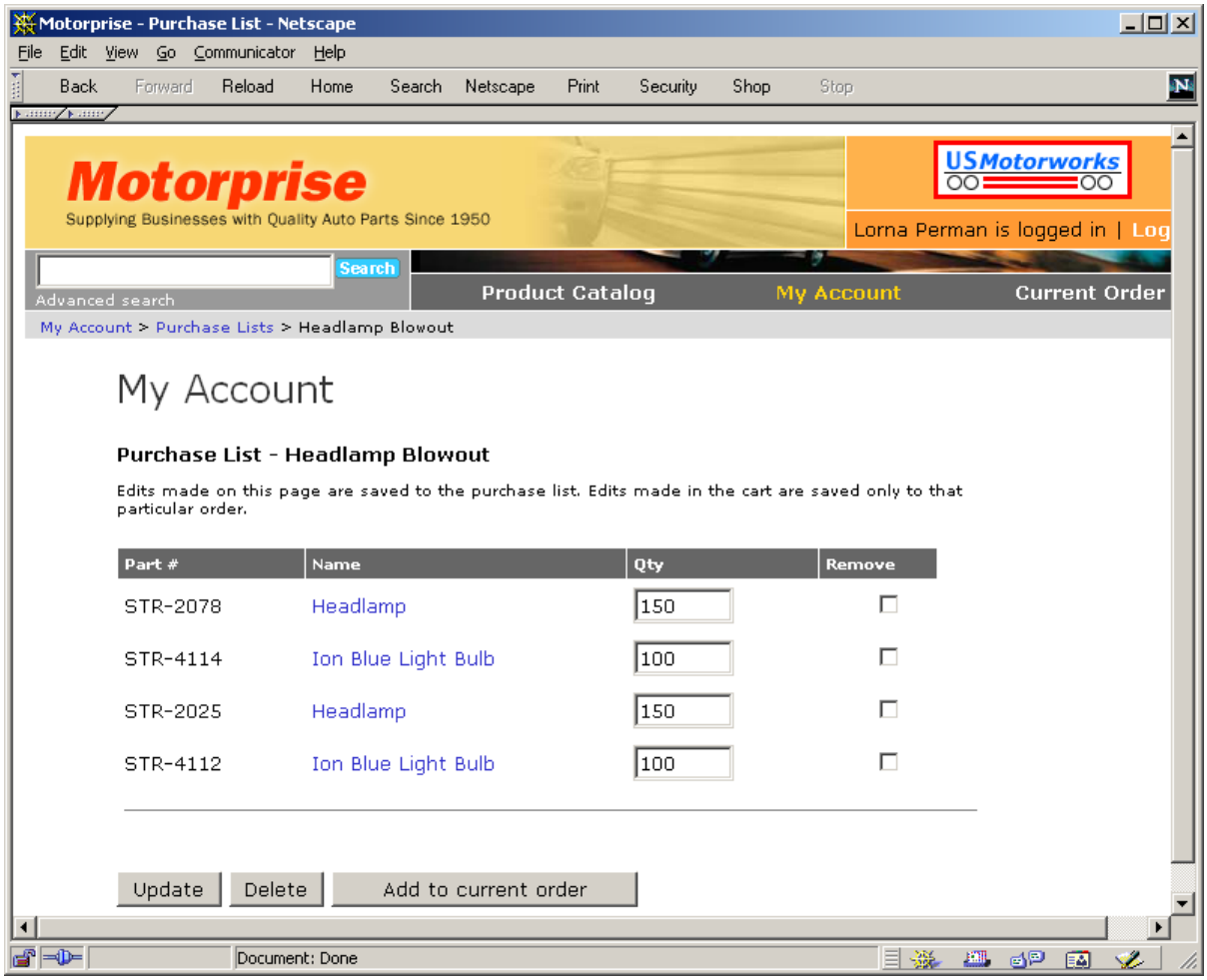
オーダー状態の詳細は、『ATG Web Commerce Programming Guide』を参照してください。

購入リストの使用

Motorprise における購入リストは、頻繁にオーダーされる品目で、一緒にオーダーされることが多い品目のリストです。顧客はこのようなリストを作成および保守することで、オーダーのたびに製品カタログから同じ品目を検索しなくてすみます。購入リストを 1 回作成しておきます。それにより、該当の品目をオーダーする場合はいつでも、購入リストを選択し、その中のすべての品目または一部の品目をオーダーに追加できます。

たとえば、バイヤーの Lorna Perman（USMW US Division - Retail）は、Headlamp Blowout という購入リストを作成しました。これは、Retail 部が定期的にセールで扱うヘッドランプと電球のセットです。

Lorna はその購入リスト内のすべての品目を現在のオーダーに追加できます。その後、カタログから追加の品目を選択することや、現在のオーダーから品目を削除することができます。



Lorna Perman の「Headlamp Blowout」購入リスト

購入リストは永続的で、バイヤー固有です。また、特定のバイヤーの各購入リストは一意の名前を持つ必要があります。Lorna が現在のオーダーに購入リストを追加した後も、購入リストはプロファイル内に引き続き存在します。そのリストを再び使用することや、「My Account」の「Purchase List」ページで後で削除することができます。

バイヤーは購入リストを編集、削除または追加できます。購入リストに対して行われた変更は、その購入リストを使用してすでに発行されたオーダーには影響しません。バイヤーが購入リストを編集すると、その変更を永続的に保存することを求めるプロンプトが表示されます。

顧客がカタログ内の製品を見ている場合（`<ATG11dir>/MotorpriseJSP/j2ee-apps/Motorprise/web-app/en/catalog/product.jsp` の表示）、新規購入リストを作成するか、表示されている製品を保存済の購入リストに追加できます。

「**Add to List**」ボタンをクリックし、既存の購入リストを選択すると、その購入リストに製品が追加されます。次のコードは、MotorpriseJSP/j22ee-apps/motorprise/web-app/en/catalog/AddToList.jsp でこの機能を実装した方法を示しています。

```
<dsp:form action="product.jsp" method="post">
  <input name="id" type="hidden" value="<dsp:valueof
    param="product.repositoryId"/>">
  <dsp:input bean="PurchaselistFormHandler.addItemToPurchaselistErrorURL"
     type="hidden" value="product.jsp"/>
  <dsp:input bean="PurchaselistFormHandler.productId"
    paramvalue="product.repositoryId" type="hidden"/>
  <dsp:droplet name="/atg/dynamo/droplet/ForEach">
    <dsp:param name="array" param="product.childSKUs"/>
    <dsp:oparam name="output">
      <table border=0 cellpadding=3 width=100%>
        <tr>
         <td><dsp:input bean="PurchaselistFormHandler.catalogRefIds"
           paramvalue="element.repositoryId" type="hidden"/>
         <span class=smallb>Qty</span> 
         <dsp:input bean="PurchaselistFormHandler.quantity" size="2" type="text"
           value="1"/> 
    </dsp:oparam>
  </dsp:droplet>

  <dsp:select bean="PurchaselistFormHandler.purchaseListId">
    <dsp:droplet name="ForEach">
      <dsp:param bean="Profile.purchaselists" name="array"/>
      <dsp:oparam name="output">
        <dsp:getvalueof id="elem" idtype="atg.repository.RepositoryItem"
          param="element">
          <dsp:option value="<%=elem.getRepositoryId()%>"/>
          <dsp:valueof param="element.eventName">Unnamed Purchase
            List</dsp:valueof>
        </dsp:getvalueof>
      </dsp:oparam>
    </dsp:droplet>
  </dsp:select></td>
  </tr>
  <tr>
    <td><dsp:input bean="PurchaselistFormHandler.addItemToPurchaselist"
      type="submit" value="Add to list"/></td>
  </tr>
<tr>
    <td>
    <table border=0 cellpadding=3 width=100%>
      <tr>
        <td><span class=smallb><dsp:a href="../user/
          purchase_lists.jsp?noCrumbs=false"><dsp:param name="product"
          param="product.repositoryId"/><dsp:param name="noCrumbs"
```

```
          value="false"/>Create new purchase list</dsp:a></span></td>
      </tr>
    </table>
    </td>
  </tr>

  </table>
</dsp:form>
```

「**Create new purchase list**」ボタンをクリックすると「Purchase Lists」ページ（user/purchase_lists.jsp）に移動し、保存済のすべての購入リストが表示されます。ここで、新規購入リストを作成できます。これを実現するために、次の JSP コードが使用されています。

```
<dsp:form action="purchase_lists.jsp" method="post">
<input name="noCrumbs" type="hidden" value="<dsp:valueof param='noCrumbs'/>">
New purchase list:<br>
<dsp:input bean="PurchaselistFormHandler.listName" maxlength="20" size="20"
   type="text"  value=""/>
      <dsp:input bean="PurchaselistFormHandler.savePurchaseList" type="hidden"
        value=""/>
<dsp:input bean="PurchaselistFormHandler.savePurchaseList" type="submit"
    value="Create list"/>
<br>
<span class="help">List names are limited to 20 characters in length</span>
</dsp:form>
```

前述のコードでは、ユーザーに入力ボックスが提供され、購入リストの名前が提示されます。その名前の購入リストがない場合は、該当の名前で購入リストが作成され、ユーザーは同じページにリダイレクトされて、新規作成された購入リストが表示されます。その名前の別の購入リストをユーザーが持っている場合は、同じページにリダイレクトされて、購入リストの名前がすでに存在することを示すエラー・メッセージが表示されます。

Motorprise では、Oracle Commerce Core Commerce のギフト・リスト機能を拡張して購入リストが作成されました。PurchaselistFormHandler を作成するためには、GiftlistFormHandler コンポーネントが拡張されました。このコンポーネントは atg/projects/b2bstore/purchaselists にあります。

PurchaselistFormHandler.properties:

```
$class=atg.projects.b2bstore.purchaselists.PurchaselistFormHandler
$scope=request
profile=/atg/userprofiling/Profile
defaultLocale^=/atg/commerce/pricing/PricingTools.defaultLocale

giftlistRepository=/atg/commerce/gifts/Giftlists
giftlistManager=/atg/commerce/gifts/GiftlistManager
orderManager=/atg/commerce/order/OrderManager
shoppingCart=/atg/commerce/ShoppingCart
giftlistTools=/atg/commerce/gifts/GiftlistTools
```

```
catalogTools=/atg/commerce/catalog/CatalogTools
profileTools=/atg/userprofiling/ProfileTools
```

purchaseLists というユーザー用の新規プロパティが作成されています。このプロパティは、イベント日、名前および出荷先所在地を必要としない点が、ギフト・リスト・プロパティとは異なります。PurchaseListFormHandler 内でこれらがすべて null に設定され、購入リストの名前のみが設定されています。

PurchaselistFormHandler は GiftlistFormHandler を拡張しています。イベントを渡す前に、購入リストにとって不要なギフト・リストの様々なプロパティを null に設定します。購入リスト名の重複もチェックします。また、GiftlistFormHandler とは異なり、PurchaselistFormHandler は、現在のオーダーから品目が移動されたときに、品目の数量を減らすことや、ギフト・リストからそれらを削除することはありません。

Motorprise では、GiftlistTools の personalGiftlistsProperty を上書きして purchaseLists をプロパティとして使用して、ユーザーの購入リストを保持しています。

オーダーのスケジュール

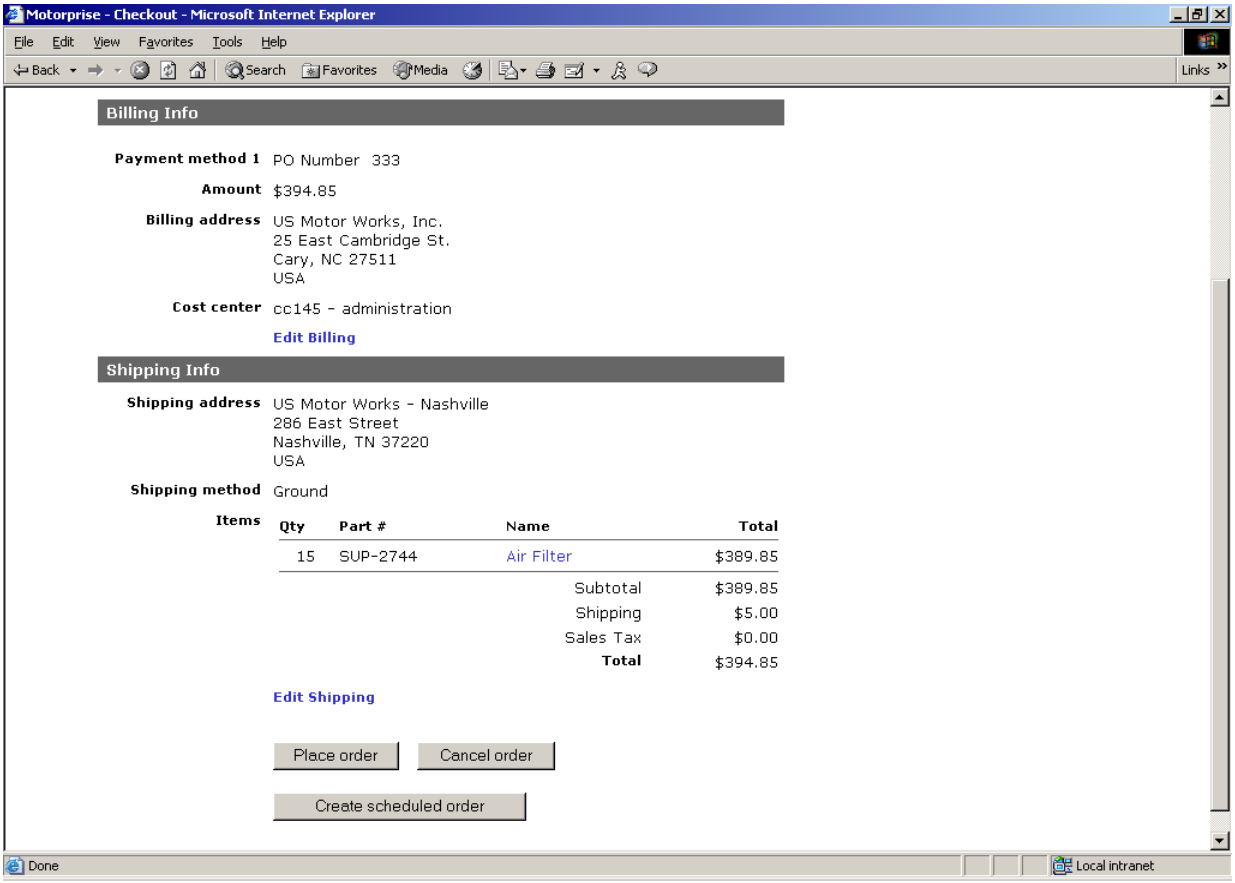
Motorprise バイヤーは、ユーザーによる操作を必要とせずに、事前に定義したスケジュールで発行される定期オーダーを作成できます。定期オーダーは、精算プロセス時または「My Account」ページを通じてスケジュールできます。

定期オーダーに承認が必要な場合は、承認保留状態で指定の間隔で承認者に自動的に送られます。承認されると、オーダーは自動的に履行されます。

定期オーダーを管理するためには、/atg/commerce/order/scheduled/ScheduledOrderFormHandler（atg.b2bcommerce.order.scheduled.ScheduledOrderHandler のインスタンス）が使用されています。この session-scoped のコンポーネントは、定期オーダーの確認、作成、更新および削除を行います。

精算時のオーダーのスケジュール

バイヤーが通常の精算処理を進める中で、オーダー確認画面の「**Place order**」ボタンではなく、「**Create scheduled order**」ボタンをクリックできます。その場合は「Scheduled Order」ページ（user/scheduled_order_new.jsp）に移動し、定期オーダーの名前とタイプ（日次、週次または月次）を指定できます。



精算時のオーダーのスケジュール。

次のコードは、定期オーダーを作成する scheduled_order_new.jsp 内のコードです。

最初に、prototypeOrderId が設定されているかどうかをチェックすることにより、オーダーの ID 番号を検索します。

```
<dsp:droplet name="IsNull">
        <dsp:param name="value" param="prototypeOrderId"/>
        <dsp:oparam name="true">
```

次に、ScheduledOrderFormHandler を使用して、ユーザーがオーダー情報を入力できるようにします。

```
<tr valign=bottom>
  <td valign="top" align=right><span class=smallb>Name</span></td>
  <td><dsp:input bean="ScheduledOrderFormHandler.value.name" name="name" size="35"
    type="text"/><br>
  <span class=help>Enter a name for your scheduled order (i.e. Weekly Spark Plug
    Assortment).</span></td>
</tr>
<tr>
  <td align=right><span class=smallb>Schedule type</span></td>
```

```
  <td align="left">
    <dsp:select bean="ScheduledOrderFormHandler.moveToMode" name="select"
      size="1">
      <dsp:option value=""/>Choose schedule
      <dsp:option value="Daily"/>daily
      <dsp:option value="Weekly"/>weekly
      <dsp:option value="Monthly"/>monthly
    </dsp:select>
  </td>
</tr>

<tr>
  <td align="right" valign=top><span class=smallb>Order</span></td>
  <td>
    <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
      templateOrderId.userInputFields.orderId">
      <dsp:droplet name="ForEach">
        <dsp:param name="array" param="result"/>
      <dsp:oparam name="outputStart">
        <dsp:option value=""/>  Select a fulfilled order
      </dsp:oparam>

      <dsp:oparam name="output">
        <dsp:getvalueof id="elementId" idtype="String" param="element.id">
          <dsp:option value="<%=elementId%>"/><dsp:valueof param="element.id"/>
        </dsp:getvalueof>
      </dsp:oparam>

      </dsp:droplet>
    </dsp:select>
<br>


  </td>
</tr>

<tr><td><img src="../images/d.gif"></td></tr>
<tr>
  <td></td>
    <input name="createNew" type="hidden" value="<dsp:valueof
      param="createNew"/>">
    <dsp:input bean="ScheduledOrderFormHandler.moveToErrorURL" type="hidden"
      value="scheduled_order_new.jsp"/>
    <dsp:input bean="ScheduledOrderFormHandler.moveToSuccessURL" type="hidden"
      value="scheduled_order_calendar.jsp?createNew=new"/>
      <td><dsp:input bean="ScheduledOrderFormHandler.moveToURL" type="submit"
        value="Continue"/></td>
</tr>
```

精算ページおよび「My Account」における定期オーダーの作成には、同じページ（scheduled_order_new.jsp）が使用されます（ユーザーが「My Account」の「Scheduled Orders」ページで新規定期オーダーを作成する場合は、ユーザーのすべての履行済オーダーのリストが表示されます。詳細は、次の「*「My Account」におけるオーダーのスケジュール*」を参照してください）。オーダー確認画面で設定される非表示のフォーム・フィールド prototypeOrderId に基づいて、精算プロセスから定期オーダーを作成するかどうかが決定されます。

オーダー確認ページからの定期オーダーの作成をユーザーが選択すると、作成されているオーダーの ID が prototypeOrderId に割り当てられ、それが scheduled_order_new.jsp に渡されます。そこからオーダーID が ScheduledOrderFormHandler.complexScheduledOrderMap.templateOrderId.userInputFields.orderId に割り当てられます。

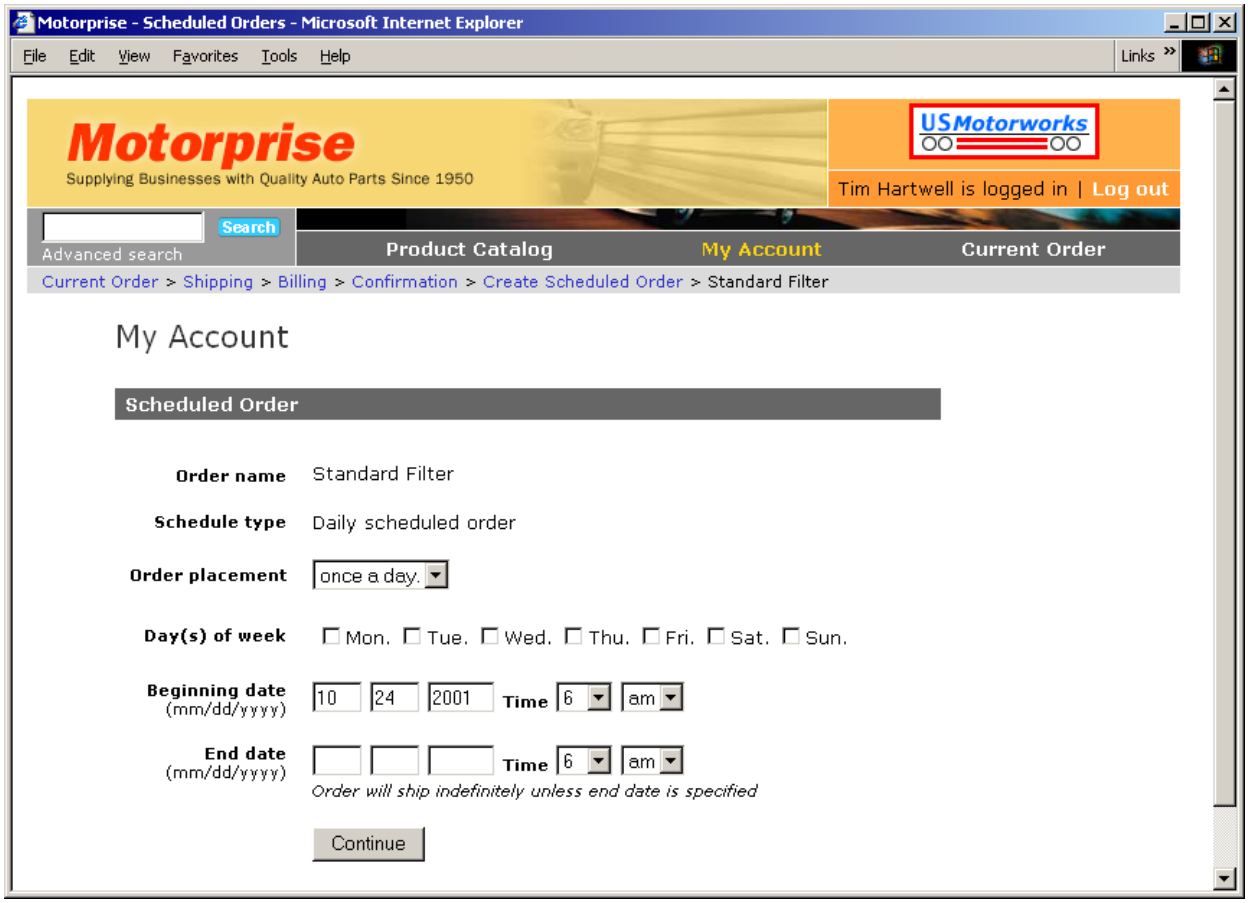
バイヤーがオーダーの名前とスケジュール・タイプを入力すると、scheduled_order_calendar.jsp に移動し、そのスケジュール・タイプ（日次、週次または月次）に応じてオーダーのカレンダが表示されます。該当のカレンダを表示するためには、次の JSP コードが使用されています。

```
<dsp:form action="scheduled_order_preview.jsp" method="post">
<table border=0 cellpadding=8>
   <tr valign=bottom>
      <td align=right><span class=smallb>Order name</span></td>
      <td><dsp:valueof bean="ScheduledOrderFormHandler.value.name" /></td>
   </tr>
   <tr>
      <td align=right><span class=smallb>Schedule type</span></td>
      <td> <dsp:valueof bean="ScheduledOrderFormHandler.moveToMode" />
         scheduled order</td>
   </tr>

<dsp:droplet name="Switch">
   <dsp:param bean="ScheduledOrderFormHandler.moveToMode" name="value"/>

   <dsp:oparam name="Daily">
      <tr>
         <td align="right"><span class=smallb>Order placement</span></td>
         <td align="left">
            <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
               calendarSchedule.userInputFields.scheduleMode" name="select"
               size="1">
               <dsp:option value="onceDaily"/>once a day.
               <dsp:option value="twiceDaily"/>twice a day.
            </dsp:select>
         </td>
      </tr>
      <%@ include file="day_of_week.jspf" %>
   </dsp:oparam>

   <dsp:oparam name="Weekly">
      <dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
         calendarSchedule.userInputFields.scheduleMode" type="hidden"
         value="Weekly"/>
```

```
   <%@ include file="day_of_week1.jspf"%>
   <tr>
      <td align="right" valign="top"><span class=smallb>Week(s) of
         month</span></td>
      <td align="left">
      <table border="0">
         <tr>
            <td><dsp:input bean="ScheduledOrderFormHandler.
               complexScheduledOrderMap.calendarSchedule.userInputFields.
               occuranceInMonth" name="occuranceInMonth" type="checkbox"
               value="1"/>1st. </td>
            <td><dsp:input bean="ScheduledOrderFormHandler.
               complexScheduledOrderMap.calendarSchedule.userInputFields.
               occuranceInMonth" name="occuranceInMonth" type="checkbox"
               value="2"/>2nd. </td>
            <td><dsp:input bean="ScheduledOrderFormHandler.
               complexScheduledOrderMap.calendarSchedule.userInputFields.
               occuranceInMonth" name="occuranceInMonth" type="checkbox"
               value="3"/>3rd. </td>
            <td><dsp:input bean="ScheduledOrderFormHandler.
            complexScheduledOrderMap.calendarSchedule.userInputFields.
            occuranceInMonth" name="occuranceInMonth" type="Checkbox"
            value="4"/>4th. </td>
            <td><dsp:input bean="ScheduledOrderFormHandler.
               complexScheduledOrderMap.calendarSchedule.userInputFields.
               occuranceInMonth" name="occuranceInMonth" type="checkbox"
               value="5"/>5th. </td>
         </tr>
         <tr><td colspan=5><span class=help>Uncheck to ship selected
            weeks</span></td></tr>
      </table>
   </tr>
</dsp:oparam>

<dsp:oparam name="Monthly">
   <tr>
      <td align="right"><span class=smallb>Order Placement</span></td>
      <td align="left">
         <dsp:select bean="ScheduledOrderFormHandler.
            complexScheduledOrderMap.calendarSchedule.userInputFields.
            scheduleMode" name="select" size="1">
            <dsp:option value="onceMonthly"/>once a month.
            <dsp:option value="biMonthly"/>every two months.
            <dsp:option value="quarterly"/>every quarter.
         </dsp:select>
      </td>
   </tr>
</dsp:oparam>
```

```
    <dsp:oparam name="default">
    </dsp:oparam>
</dsp:droplet>
```

前述のコード・ブロックは、ユーザーが選択して ScheduleOrderFormHandler.moveToMode に設定された該当のカレンダを表示するための制御フローを示しています。スケジュール・タイプには、日次、週次および月次の 3 種類があります。各スケジュール・タイプのカレンダについて、次に説明します。



日次の定期オーダーを作成するカレンダ・ページ

日次スケジュールのコードを、次に示します。

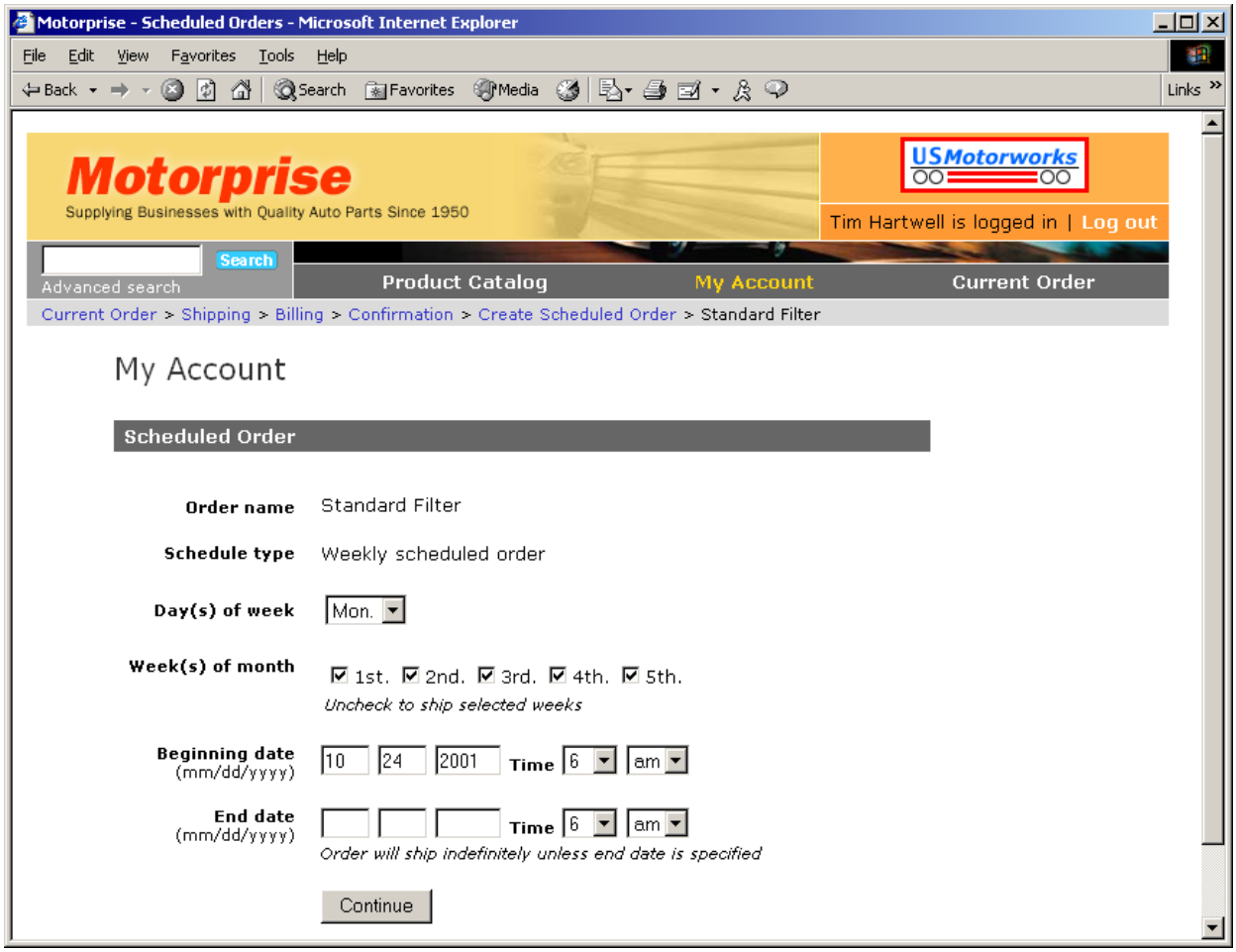
```
<dsp:oparam name="Daily">
   <tr>
      <td align="right"><span class=smallb>Order placement</span></td>
      <td align="left">
         <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
            calendarSchedule.userInputFields.scheduleMode" name="select"
            size="1">
            <dsp:option value="onceDaily"/>once a day.
```

```
            <dsp:option value="twiceDaily"/>twice a day.
          </dsp:select>
        </td>
    </tr>
    <%@ include file="day_of_week.jspf" %>
</dsp:oparam>
```

day_of_week.jspf ページ・フラグメントには、次に示すように、曜日を表示し、それを complexScheduledOrderMap に割り当てるコードが含まれています。

```
dsp:importbean bean="/atg/commerce/order/scheduled/ScheduledOrderFormHandler"/>
<!-- Selection of days for shipment including x times/week -->
<tr>
  <td align="right"><span class=smallb>Day(s) of week</span></td>
  <td align="left">
    <table>
      <tr>
        <td><dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
           calendarSchedule.userInputFields.daysOfWeek" name="daysOfWeek"
           type="checkbox" value="2"/>Mon. </td>
        <td><dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
           calendarSchedule.userInputFields.daysOfWeek" name="daysOfWeek"
           type="checkbox" value="3"/>Tue. </td>
        <td><dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
           calendarSchedule.userInputFields.daysOfWeek" name="daysOfWeek"
           type="checkbox" value="4"/>Wed. </td>
        <td><dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
           calendarSchedule.userInputFields.daysOfWeek" name="daysOfWeek"
           type="Checkbox" value="5"/>Thu. </td>
        <td><dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
           calendarSchedule.userInputFields.daysOfWeek" name="daysOfWeek"
              type="checkbox" value="6"/>Fri. </td>
        <td><dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
           calendarSchedule.userInputFields.daysOfWeek" name="daysOfWeek"
              type="Checkbox" value="7"/>Sat. </td>
        <td><dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
           calendarSchedule.userInputFields.daysOfWeek" name="daysOfWeek"
           type="Checkbox" value="1"/>Sun. </td>
     </tr>
  </table>
</tr>
</dsp:page>
```

前述のように、日次スケジュールのオプションとして「once a day」と「twice a day」、および曜日と対応するチェック・ボックスが表示され、それぞれを選択できます。



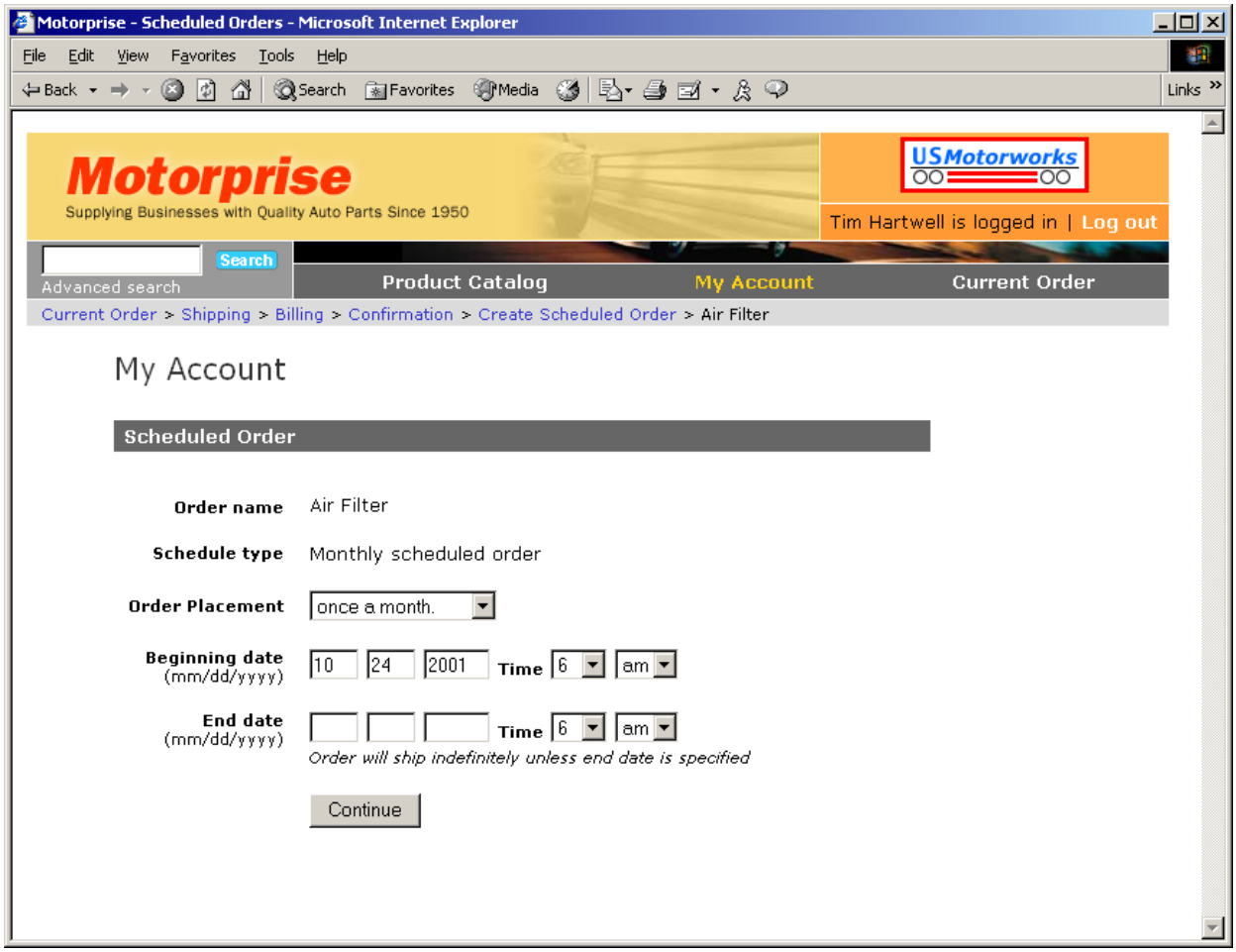
週次の定期オーダーを作成するカレンダ・ページ

週次スケジュールのコードを、次に示します。

```
<dsp:oparam name="Weekly">
   <dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
      calendarSchedule.userInputFields.scheduleMode" type="hidden"
         value="Weekly"/>
   <%@ include file="day_of_week1.jspf"%>
   <tr>
      <td align="right" valign="top"><span class=smallb>Week(s) of
         month</span></td>
      <td align="left">
      <table border="0">
         <tr>
            <td><dsp:input bean="ScheduledOrderFormHandler.
               complexScheduledOrderMap.calendarSchedule.userInputFields.
               occuranceInMonth" name="occuranceInMonth" type="checkbox"
```

```
                value="1"/>1st. </td>
             <td><dsp:input bean="ScheduledOrderFormHandler.
                complexScheduledOrderMap.calendarSchedule.userInputFields.
                occuranceInMonth" name="occuranceInMonth" type="checkbox"
                value="2"/>2nd. </td>
             <td><dsp:input bean="ScheduledOrderFormHandler.
                complexScheduledOrderMap.calendarSchedule.userInputFields.
                occuranceInMonth" name="occuranceInMonth" type="checkbox"
                value="3"/>3rd. </td>
             <td><dsp:input bean="ScheduledOrderFormHandler.
                complexScheduledOrderMap.calendarSchedule.userInputFields.
                occuranceInMonth" name="occuranceInMonth" type="Checkbox"
                value="4"/>4th. </td>
            <td><dsp:input bean="ScheduledOrderFormHandler.
               complexScheduledOrderMap.calendarSchedule.userInputFields.
               occuranceInMonth" name="occuranceInMonth" type="checkbox"
               value="5"/>5th. </td>
          </tr>
          <tr><td colspan=5><span class=help>Uncheck to ship selected
             weeks</span></td></tr>
       </table>
    </tr>
</dsp:oparam>
```

前述のコードでは ScheduledOrderFormHandler.scheduledMode を Weekly に設定し、ユーザーが曜日を選択できるように曜日を選択ボックスに表示します。月内での週の番号もすべて表示され、対応するチェック・ボックスで、オーダーをスケジュールする週を選択できます。



月次の定期オーダーを作成するカレンダ・ページ

次のコードは、月次スケジュールを表示するコードです。

```
<dsp:oparam name="Monthly">
   <tr>
      <td align="right"><span class=smallb>Order Placement</span></td>
      <td align="left">
         <dsp:select bean="ScheduledOrderFormHandler.
            complexScheduledOrderMap.calendarSchedule.userInputFields.
            scheduleMode" name="select" size="1">
            <dsp:option value="onceMonthly"/>once a month.
            <dsp:option value="biMonthly"/>every two months.
            <dsp:option value="quarterly"/>every quarter.
         </dsp:select>
      </td>
   </tr>
</dsp:oparam>
```

月次スケジュールでユーザーが選択できるモードは、「once a month」、「every two months」および「every quarter」の 3 種類です。

最後に、各カレンダに対して、開始日と終了日のフィールドを表示します。start_end_date.jspf からの次のコード・スニペットは scheduled_order_new.jsp 内で起動されて、開始日と終了日のオプションを表示します。

```
<tr valign=top>
  <td align=right><span class=smallb>Beginning date</span><br>
  <span class="small">(mm/dd/yyyy)</span></td>
  <td><dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
    startDate.userInputFields.month" maxlength="2" size="2" type="text"/>
  <dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
    startDate.userInputFields.day" maxlength="2" size="2" type="text"/>
  <dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
    startDate.userInputFields.year" maxlength="4" size="4" type="text"/>
  <span class=smallb>Time</span>

  <dsp:droplet name="Switch">
  <dsp:param bean="Profile.locale" name="value"/>
  <dsp:oparam name="en_US">
    <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
      startDate.userInputFields.hour">
    <dsp:droplet name="For">
      <dsp:param name="howMany" value="12"/>
      <dsp:oparam name="output">
        <dsp:getvalueof id="numHoursEn" idtype="Integer" param="count">
          <dsp:option value="<%=numHoursEn.toString()%>"/> <dsp:valueof
            param="count"/>
        </dsp:getvalueof>
      </dsp:oparam>
    </dsp:droplet>
    </dsp:select>

    <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
      startDate.userInputFields.ampm">
     <dsp:option value="am"/> am
     <dsp:option value="pm"/> pm
    </dsp:select>
  </dsp:oparam>
  <dsp:oparam name="de_DE">
    <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
      startDate.userInputFields.hour">
    <dsp:droplet name="For">
      <dsp:param name="howMany" value="24"/>
      <dsp:oparam name="output">
        <dsp:getvalueof id="numHoursDe" idtype="Integer" param="count">
          <dsp:option value="<%= numHoursDe.toString() %>"/> <dsp:valueof
            param="count"/>
        </dsp:getvalueof>
```

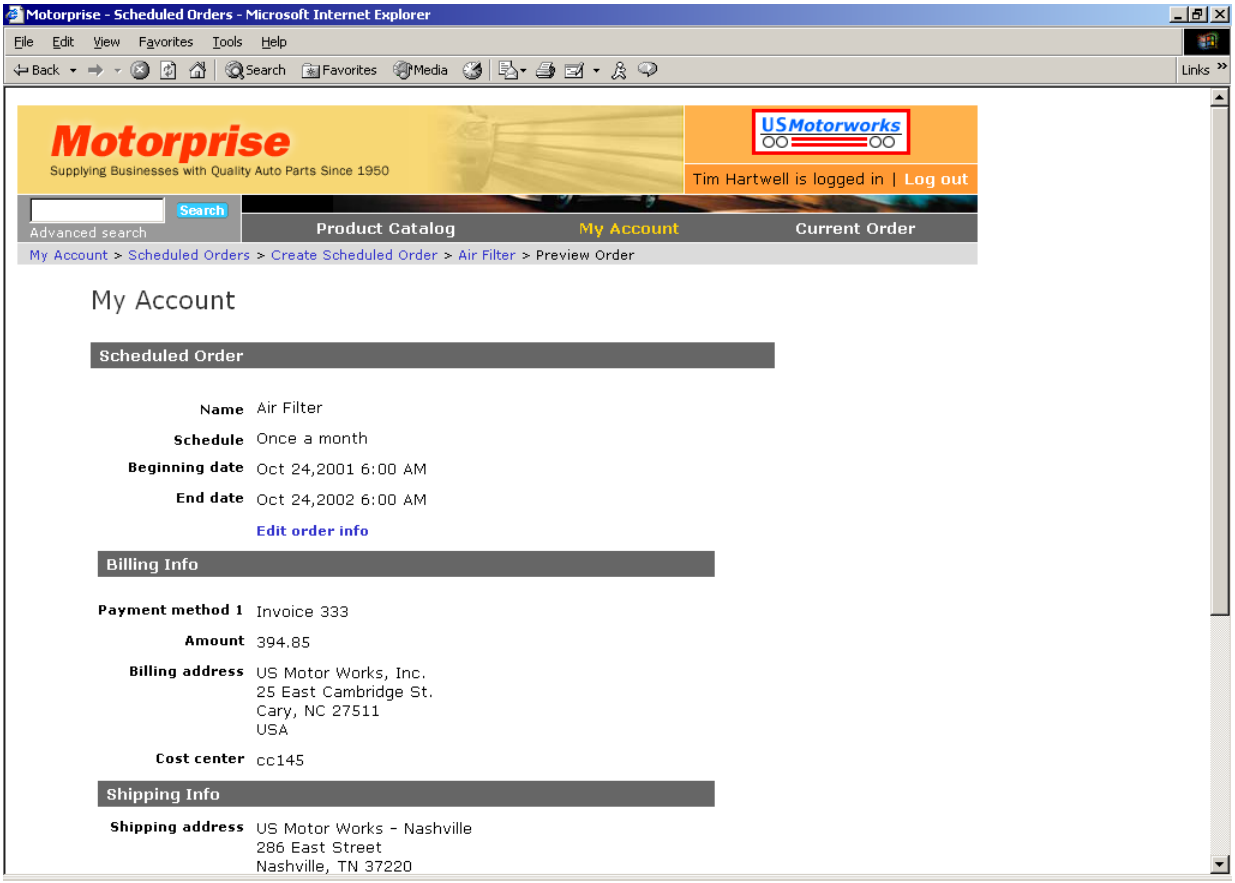
```
      </dsp:oparam>
    </dsp:droplet>
    </dsp:select>
  </dsp:oparam>
  </dsp:droplet>
  </td>
</tr>


<tr valign=top>
     <!--<dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
       endDate.userInputFields.mode" type="hidden" value="definite"/>-->
  <td align=right><span class=smallb>End date</span><br>
  <span class="small">(mm/dd/yyyy)</span></td>
  <td><dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
    endDate.userInputFields.month" maxlength="2" size="2" type="text"/>
  <dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
    endDate.userInputFields.day" maxlength="2" size="2" type="text"/>
  <dsp:input bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
    endDate.userInputFields.year" maxlength="4" size="4" type="text"/>
  <span class=smallb>Time</span>

  <dsp:droplet name="Switch">
  <dsp:param bean="Profile.locale" name="value"/>
  <dsp:oparam name="en_US">
    <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
      endDate.userInputFields.hour">
    <dsp:droplet name="For">
      <dsp:param name="howMany" value="12"/>
        <dsp:oparam name="output">
        <dsp:getvalueof id="hourEndDe" idtype="Integer" param="count">
          <dsp:option value="<%=hourEndDe.toString()%>"/> <dsp:valueof
            param="count"/>
        </dsp:getvalueof>

        </dsp:oparam>
   </dsp:droplet>
   </dsp:select>

   <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.endDate.
     userInputFields.ampm">
    <dsp:option value="am"/> am
    <dsp:option value="pm"/> pm
   </dsp:select>
  </dsp:oparam>
  <dsp:oparam name="de_DE">
    <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
      endDate.userInputFields.hour">
    <dsp:droplet name="For">
```

```
      <dsp:param name="howMany" value="24"/>
      <dsp:oparam name="output">
        <dsp:getvalueof id="hourEndEn" idtype="Integer" param="count">
          <dsp:option value="<%=hourEndEn.toString()%>"/> <dsp:valueof
            param="count"/>
        </dsp:getvalueof>
      </dsp:oparam>
    </dsp:droplet>
    </dsp:select>
  </dsp:oparam>
  </dsp:droplet>
```

提供されているいずれかのオプションをユーザーが選択し、オーダーをプレビューできるように、発行ボタンを表示します。次のコードは、scheduled_order_new.jsp からの関連コードです。

```
<tr>
  <td></td>
  <input name="createNew" type="hidden" value="<dsp:valueof param="createNew"/>">
  <dsp:input bean="ScheduledOrderFormHandler.moveToErrorURL" type="hidden"
    value="scheduled_order_new.jsp"/>
  <dsp:input bean="ScheduledOrderFormHandler.moveToSuccessURL" type="hidden"
    value="scheduled_order_calendar.jsp?createNew=new"/>
  <td><dsp:input bean="ScheduledOrderFormHandler.moveToURL" type="submit"
    value="Continue"/></td>
</tr>
```

ユーザーがカレンダのオプションを選択し、「**Continue**」ボタンをクリックすると、scheduled_order_preview.jsp に移動し、オーダー情報、および定期オーダーを作成する発行ボタンが表示されます。



定期オーダーのプレビュー

次のコードは、定期オーダーを作成または削除する発行ボタンを表示する、scheduled_order_preview.jsp からのコードです。

```
<dsp:droplet name="IsNull">
  <dsp:param name="value" bean="ScheduledOrderFormHandler.repositoryId"/>
  <dsp:oparam name="true">
    <dsp:input bean="ScheduledOrderFormHandler.createErrorURL" type="hidden"
      value="scheduled_order_new.jsp"/>
    <dsp:input bean="ScheduledOrderFormHandler.createSuccessURL" type="hidden"
      value="scheduled_orders.jsp"/>
    <dsp:input bean="ScheduledOrderFormHandler.create" type="submit" value="Create
      scheduled order"/>
  </dsp:oparam>
  <dsp:oparam name="false">
    <dsp:input bean="ScheduledOrderFormHandler.deleteSuccessURL" type="hidden"
      value="scheduled_orders.jsp"/>
    <dsp:input bean="ScheduledOrderFormHandler.deleteErrorURL" type="hidden"
      value="scheduled_order_preview.jsp"/>
    <dsp:input bean="ScheduledOrderFormHandler.delete" type="submit" value="Delete
```

```
      scheduled order"/><p>
    <span class=smallb><dsp:a href="scheduled_orders.jsp">Return to scheduled
      orders</dsp:a></span>
  </dsp:oparam>
</dsp:droplet>
```

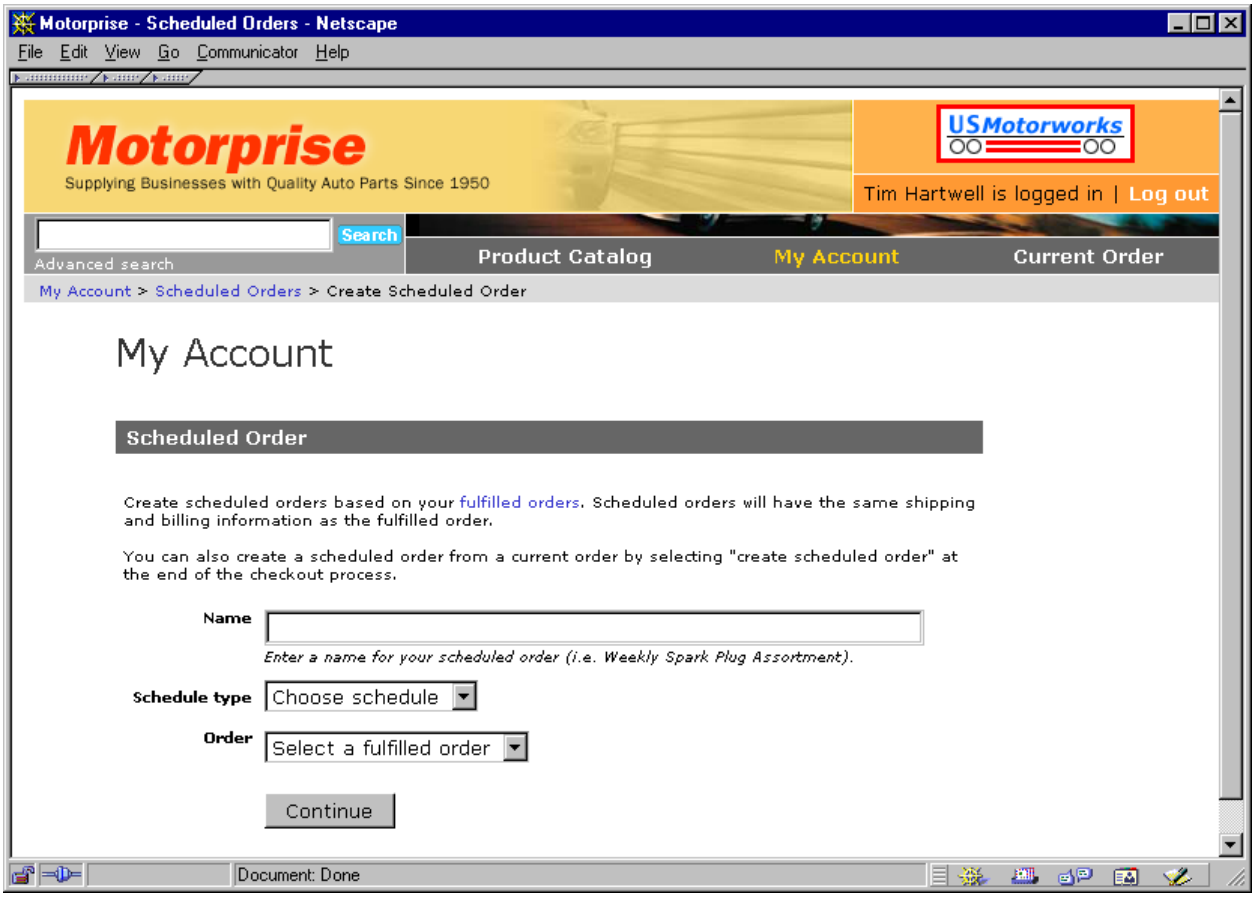
ユーザーは scheduled_orders.jsp に移動し、そこですべての定期オーダーが実際に表示されます。次のコードは、特定のユーザーの定期オーダーをフェッチして表示するコードです。

```
<dsp:droplet name="RQLQueryForEach">
   <dsp:param bean="/atg/dynamo/transaction/TransactionManager"
      name="transactionManager"/>
   <dsp:param bean="/atg/commerce/order/OrderRepository" name="repository"/>
   <dsp:param name="itemDescriptor" value="scheduledOrder"/>
   <dsp:param name="queryRQL" value="profileId=:profileId"/>
   <dsp:param bean="Profile.repositoryId" name="profileId"/>
   <dsp:oparam name="output">
      <dsp:a href="scheduled_order_preview.jsp">
      <dsp:param name="scheduledOrderId" param="element.repositoryId"/>
      <dsp:param name="source" value="scheduledOrder"/>
      <dsp:valueof param="element.name"/>
      </dsp:a><br>
   </dsp:oparam>
   <dsp:oparam name="empty">
      You have no scheduled orders.
   </dsp:oparam>
</dsp:droplet>
```

特定のユーザーの定期オーダーをフェッチし、各オーダーの scheduled_order_preview.jsp へのリンクを提供するためには、RQLQueryForEach サーブレット Bean が使用されています。各定期オーダーのリポジトリ ID は scheduledOrderId パラメータに割り当てられ、ページに渡されます。詳細は、『ATG Web Commerce Page Developer's Guide』の RQLQueryForEach サーブレット Bean に関する説明を参照してください。

「My Account」でのオーダーのスケジュール

「My Account」の「Scheduled Orders」ページ（user/scheduled_orders.jsp）で、「**Create scheduled order**」リンクをクリックして定期オーダーを作成することもできます。



「My Account」での定期オーダーの設定。

「My Account」からの定期オーダーの作成は、オーダー確認ページからの作成と似ています。ただし、テンプレート・オーダーが `ScheduledOrderFormHandler` に渡される点が異なります。オーダー確認ページからの定期オーダーの作成をユーザーが選択すると、作成されているオーダーの ID が `prototypeOrderId` に割り当てられ、それが `scheduled_order_new.jsp` に渡され、そこからオーダーID が `ScheduledOrderFormHandler.complexScheduledOrderMap.templateOrderId. userInputFields.orderId` に割り当てられます。ユーザーが「My Account」の「Scheduled Orders」ページで**「Create new scheduled order」**リンクをクリックすると、ユーザーのすべての履行済オーダーのリストが表示されます。ユーザーは、`ScheduledOrderFormHandler.complexScheduledOrderMap.templateOrderId. userInputFields.orderId` に割り当てられているこのリストからオーダーを選択できます。

次のコードは、「My Account」から定期オーダーを作成するための履行済オーダーおよび発行ボタンを表示する、`scheduled_order_new.jsp` 内のコードです。

```
<dsp:droplet name="OrderLookup">
   <dsp:param name="userId" bean="Profile.repositoryid"/>
   <dsp:param name="state" value="closed"/>
   <dsp:oparam name="output">
      <dsp:setvalue param="result" paramvalue="result"/>
   </dsp:oparam>
```

```
</dsp:droplet>

<dsp:droplet name="IsNull">
   <dsp:param name="value" param="prototypeOrderId"/>
   <dsp:oparam name="true">
      <dsp:droplet name="OrderLookup">
         <dsp:param name="userId" bean="Profile.repositoryId"/>
         <dsp:param name="state" value="closed"/>
            <dsp:oparam name="output">
               <dsp:form action="scheduled_order_new.jsp" method="post">

<table border=0 cellpadding=3 width=100%>
   <tr>
      <td colspan=2>
         <dsp:droplet name="IsNull">
            <dsp:param bean="ScheduledOrderFormHandler.repositoryId"
               name="value"/>
            <dsp:oparam name="true"><span class=small>Create scheduled orders
               based on your <dsp:a href="orders_filled.jsp">fulfilled
               orders</dsp:a>. Scheduled orders will have the same shipping and
               billing information as the fulfilled order. <p>You can also create
               a scheduled order from a current order by selecting "create
               scheduled order" at the end of the checkout process.</span>
            </dsp:oparam>
            <dsp:oparam name="false"><span class=small>You can edit the name of
               your order, schedule type, and order placement information.</span>
            </dsp:oparam>
         </dsp:droplet>
      </td>
   </tr>
   <tr><td><dsp:img src="../images/d.gif"/></td></tr>
   <tr valign=bottom>
      <td valign="top" align=right><span class=smallb>Name</span></td>
      <td><dsp:input bean="ScheduledOrderFormHandler.value.name" name="name"
         size="35" type="text"/><br>
      <span class=help>Enter a name for your scheduled order (i.e. Weekly Spark
         Plug Assortment).</span></td>
   </tr>
   <tr>
      <td align=right><span class=smallb>Schedule type</span></td>
      <td align="left">
         <dsp:select bean="ScheduledOrderFormHandler.moveToMode" name="select"
            size="1">
            <dsp:option value=""/>Choose schedule
            <dsp:option value="Daily"/>daily
            <dsp:option value="Weekly"/>weekly
            <dsp:option value="Monthly"/>monthly
         </dsp:select>
      </td>
```

```
   </tr>

<dsp:droplet name="IsNull">
   <dsp:param bean="ScheduledOrderFormHandler.repositoryId" name="value"/>
   <dsp:oparam name="true">
   <tr>
      <td align="right" valign=top><span class=smallb>Order</span></td>
      <td>
         <dsp:select bean="ScheduledOrderFormHandler.complexScheduledOrderMap.
            templateOrderId.userInputFields.orderId">
            <dsp:droplet name="ForEach">
               <dsp:param name="array" param="result"/>
               <dsp:oparam name="outputStart">
                  <dsp:option value=""/>  Select a fulfilled order
               </dsp:oparam>
               <dsp:oparam name="output">
                  <dsp:getvalueof id="option384" param="element.id"
                     idtype="java.lang.String">
                     <dsp:option value="<%=option384%>"/>
                  </dsp:getvalueof><dsp:valueof param="element.id"/>
               </dsp:oparam>
            </dsp:droplet>
         </dsp:select><br>
      </td>
   </tr>
   </dsp:oparam>
   <dsp:oparam name="false">
   </dsp:oparam>
</dsp:droplet>

   <tr><td><dsp:img src="../images/d.gif"/></td></tr>
   <tr>
      <td></td>
      <input name="createNew" type="hidden" value='<dsp:valueof
         param="createNew"/>'>
      <dsp:input bean="ScheduledOrderFormHandler.moveToErrorURL" type="hidden"
         value="scheduled_order_new.jsp"/>
      <dsp:input bean="ScheduledOrderFormHandler.moveToSuccessURL" type="hidden"
         value="scheduled_order_calendar.jsp?createNew=new"/>
      <td><dsp:input bean="ScheduledOrderFormHandler.moveToURL" type="submit"
         value="Continue"/></td>
   </tr>
   <!-- vertical space -->
   <tr><td><dsp:img vspace="0" src="../images/d.gif"/></td></tr>
</table>
</dsp:form>
</dsp:oparam>
<dsp:oparam name="empty">
```

```
<dsp:param name="value" param="result"/>
<dsp:oparam name="true">
   You have no <dsp:a href="orders_filled.jsp">fulfilled orders</dsp:a> at this
   time to create a scheduled order.  <p>You can create a scheduled order from a
   current order by selecting "create scheduled order" at the end of the checkout
   process.
</dsp:oparam>
   You have no <dsp:a href="orders_filled.jsp">fulfilled orders</dsp:a> at this
   time to create a scheduled order.  <p>You can create a scheduled order from a
   current order by selecting "create scheduled order" at the end of the checkout
   process.

</dsp:oparam>
</dsp:droplet><!--end OrderLookup-->
```

前述のように、OrderLookup コンポーネントを使用して、オーダーの状態が「closed」であるすべてのユーザー・オーダーをフェッチし、それらを表示してユーザーがその中から選択できるようにしています。ユーザーがスケジュール・タイプ、定期オーダー名およびオーダーを選択すると、schedule_order_calendar.jsp に移動し、カレンダを選択します。残りのプロセスは、精算プロセスから定期オーダーを作成する場合と同じです。

定期オーダーの削除

「My Account」の「Scheduled Orders」ページで定期オーダーを選択し、ページの末尾にある**「Delete scheduled order」**ボタンをクリックすることによって、定期オーダーを削除できます。ユーザーは scheduled_order_preview.jsp に移動し、選択したオーダーのオーダーID が scheduledOrderId として渡されます。オーダー情報が表示され、オーダーを編集または削除するオプションが提供されます。scheduledOrderId は ScheduledOrderFormHandler.repositoryId に割り当てられます。

定期オーダーを削除する scheduled_order_preview.jsp からの JSP コードを、次に示します。

```
<dsp:input bean="ScheduledOrderFormHandler.deleteSuccessURL" type="hidden"
   value="scheduled_orders.jsp"/>
<dsp:input bean="ScheduledOrderFormHandler.deleteErrorURL" type="hidden"
   value="scheduled_order_preview.jsp"/>
<dsp:input bean="ScheduledOrderFormHandler.delete" type="submit" value="Delete
   scheduled order"/>
```

定期オーダーの更新

「My Account」の「Scheduled Orders」ページで定期オーダーを選択し、**「Edit order info」**リンクをクリックします。その定期オーダーの任意の情報を変更できます。必要な変更を行った後、**「Update」**ボタンをクリックします。このプロセスは、オーダーを更新するオプションと既存の定期オーダー情報を表示する点を除いて、新規定期オーダーを作成するプロセスと似ています。

まず、ユーザー入力の妥当性を確認するために、scheduled_order_calendar から次の JSP コードが使用されています。

```
<dsp:input bean="ScheduledOrderFormHandler.verifySuccessURL" type="hidden"
   value="scheduled_order_preview.jsp?source=scheduledOrder&createNew=new"/>
```

```
<dsp:input bean="ScheduledOrderFormHandler.verifyErrorURL" type="hidden"
   value="scheduled_order_calendar.jsp"/>
<dsp:input bean="ScheduledOrderFormHandler.verify" type="submit"
   value="Continue"/>
```

定期オーダーを更新するためには、次の JSP コード・スニペットが使用されています。

```
<dsp:input bean="ScheduledOrderFormHandler.updateSuccessURL" type="hidden"
   value="scheduled_orders.jsp"/>
<dsp:input bean="ScheduledOrderFormHandler.updateErrorURL" type="hidden"
   value="scheduled_order_calendar.jsp"/>
<dsp:input bean="ScheduledOrderFormHandler.update" type="submit"
   value="Update"/><p>
<span class=smallb><dsp:a href="scheduled_orders.jsp">Return to scheduled
   orders</dsp:a></span>
```

ユーザーが「**Preview**」ボタンをクリックすると、検証が実行されます。オーダー名の不足など、無効なデータが見つかると、画面の上部にエラー・メッセージが表示されます。

Oracle Commerce Core Commerce には、オーダー・リポジトリを定期的にポーリングし、そのスケジュールに応じて定期オーダーを発行する、`ScheduledOrderService` というバックエンド・サービスがあります。これは、`placeScheduledOrders` タスクを実行する頻度を定義する、`schedule` というプロパティを持ちます。Motorprise はデモなので、定期オーダーを頻繁にチェックすることが望ましいと考えて、`<ATG11dir>\MotorpriseJSP\j2ee-apps\motorprise\config\atg\commerce\order\scheduled\ScheduledOrderService.properties` 内の `schedule` プロパティは「10 秒ごと」に設定されているため、定期オーダー機能を即時に確認できます。

詳細は、『ATG Web Commerce Programming Guide』を参照してください。

オーダーの保存

バイヤーは、選択されているオーダーをセッション間で保存できます。この機能は、保存された複数のショッピング・カートを持つという概念に相当します。

`checkout/cart.jsp` および `checkout/save_order.jsp` により、ユーザーは自分が指定する摘要の元でオーダーを保存できます。

「Current Order」ページ（`checkout/cart.jsp`）では、バイヤーが精算、オーダーの取消しまたはオーダーの保存を行うことができます。「**Save Order**」ボタンをクリックすると `save_order.jsp` に移動し、保存されるオーダーの摘要の入力が求められます。ユーザーが名前を入力しない場合は、ユーザーのロケールに応じて、現在の日時スタンプで保存されます。

ユーザーが指定する名前に基づいてユーザーの現在のオーダーを保存するためには、`/atg/commerce/order/purchase/SaveOrderFormHandler` が使用されています。特定のバイヤーの保存された各オーダーは、一意の名前を持つ必要があります。保存されたオーダーのリスト内にすでに存在する名前を入力すると、エラーが表示されます。`user/save_order.jsp` 内の次の JSP コード・ブロックを使用して、これが実現されています。

```
<dsp:form action="saved_orders.jsp">
   Enter a name to identify this order:<p>
   <dsp:input bean="SaveOrderFormHandler.description" type="text"/>
   <dsp:input bean="SaveOrderFormHandler.saveOrder" type="submit" value="Save
      order"/>
   <dsp:input bean="SaveOrderFormHandler.saveOrder" type="hidden" value="save"/>
   <dsp:input bean="SaveOrderFormHandler.saveOrderSuccessURL" type="hidden"
      value="../user/saved_orders.jsp"/>
   <dsp:input bean="SaveOrderFormHandler.saveOrderErrorURL" type="hidden"
     value="../user/save_order.jsp"/>
```

ユーザーの「My Account」ページ（order_management.jsp）には、保存されたオーダーのリンクが表示されます。このリンクをクリックすると saved_orders.jsp が表示されて、リストを参照できます。このページでは、保存された任意のオーダーを現在のオーダーとして選択できます。

保存されたオーダーのリストを表示するコードを、次に示します。

```
<dsp:droplet name="IsEmpty">
  <dsp:param name="value" bean="ShoppingCart.saved"/>
  <dsp:oparam name="true">
    <tr><td>You have no saved orders.</td></tr>
  </dsp:oparam>
  <dsp:oparam name="false">
    <tr valign=top>
      <td>
      <table border=0 cellpadding=4 width=100%>
        <tr>
          <td><b><span class=small> Order name</span></b></td>
          <td>  </td>
          <td><b><span class=small> Order #</span></b></td>
          <td>  </td>
          <td><b><span class=small>Date saved</span></b></td>
        </tr>
        <dsp:droplet name="ForEach">
          <dsp:param name="array" bean="ShoppingCart.saved"/>
          <dsp:oparam name="output">
            <tr>
              <td><dsp:a href="saved_order.jsp"><dsp:param name="orderId"
                param="element.id"/><dsp:valueof
                param="element.description"/></dsp:a></td>
              <td></td>
              <td><dsp:valueof  param="element.id"/></td>
              <td></td>
              <td><dsp:valueof date="MMMM d, yyyy h:mm a" param=
               "element.creationDate"/></td>
            </tr>
          </dsp:oparam>
        </dsp:droplet>
      </table>
```

```
      </td>
    </tr>
  </dsp:oparam>
</dsp:droplet><!--end IsEmpty-->
```

ユーザーの保存されたオーダーはすべて、ショッピング・カートの saved 配列プロパティに格納されています。これを反復して保存された各オーダーとリンクを表示することで、ユーザーは保存されたオーダーにアクセスし、それを現在のショッピング・カートとして選択できます。次の saved_order.jsp からのコードは、これを示しています。

```
<td colspan=4 align=right>
   <dsp:form action="../checkout/cart.jsp" method="post">
   <dsp:input bean="ShoppingCart.handlerOrderId" paramvalue="order.id"
      type="hidden"/>
   <dsp:input bean="ShoppingCart.switch" type="submit" value="Make this your
      current order"/>  
   </dsp:form>
</td>
<td colspan=2 align=left>
   <dsp:form action="../user/saved_orders.jsp" method="post">
   <dsp:input bean="ShoppingCart.handlerOrderId" paramvalue="order.id"
      type="hidden"/>
   <dsp:input bean="ShoppingCart.delete" type="submit" value="Delete"/>  
   </dsp:form>
```

保存されたオーダーが現在のオーダーになると、そのオーダーは保存されたオーダーのリストから削除されます。

注意: 保存されたオーダーは購入リストとは異なります。購入リストはカタログをナビゲートしながら作成できます。カタログの参照中に、製品を購入リストに追加することや、新規購入リストを作成して品目を追加することができます。購入リストは完全に編集可能です。購入リストに対して品目を追加または削除するほかに、購入リスト全体を削除することもできます。購入リストは何度でも使用できます。購入リストが現在のオーダーに追加されると、購入リスト内のすべての品目が実際にオーダーに追加されます。オーダーに追加された後、購入リスト自体は削除されません。しかし、保存されたオーダーは、現在のオーダーになるときに 1 回のみ使用できます。保存されたオーダーは実際に精算プロセスに送られます。現在のオーダーになると、保存されたオーダーのリストから削除されリストに表示されなくなります。

オーダーの承認

Motorprise は、Core Commerce を使用して独自のカスタム承認条件を作成する方法を示しています。ユーザーがオーダーを発行するときに確認のために発生する承認検証プロセスには、2 つの手順が含まれます。

1. オーダーを発行するユーザーの approvalRequired プロパティをチェックして、ユーザーが承認検証プロセス全体を経由する必要があるかどうかを決定します。

   たとえば、NDAP のバイヤーである Louis Veloso からのオーダーは承認が不要なため、Louis からのオーダーについては承認条件がチェックされません。しかし、Nicole Hsu（USMW – US Division - Wholesale）のオーダーには承認が必要です。彼女の

approvalRequired プロパティは true に設定されているため、彼女が発行する任意のオーダーについて承認条件がチェックされます。

2. ユーザーの approvalRequired プロパティが true の場合、そのユーザーが発行する任意のオーダーのすべての承認条件をチェックします。独自の承認条件を作成することで、承認の動作をカスタマイズできます。Motorprise では、オーダーできる最大金額を制限するといった条件が作成されています。

   たとえば、USMW の Tim Hartwell のオーダー限度額は$10,000 です。Tim が総額$10,000 を超えるオーダーを発行する場合は、USMW で彼の部署の承認者である Mary Granger の承認を受ける必要があります。

承認条件

Core Commerce では、ユーザーの approvalRequired プロパティがデフォルトでチェックされます。これが true の場合、そのユーザーが発行するすべてのオーダーは承認が必要な状態に設定されます。Motorprise では、ユーザーの approvalRequired プロパティが true の場合に orderPriceLimit プロパティをチェックする条件を追加することで、この動作を上書きしています。

Motorprise のこの新しい承認条件は、クラス atg.commerce.expression.ProcPropertyRestriction に基づく新規コンポーネント/atg/commerce/approval/processor/CheckOrderLimitApprovalRequirements を作成することで実装されています。このコンポーネントは、ユーザーが発行したオーダーの合計額がユーザーの orderPriceLimit を超えるかどうかを確認します。

ProcPropertyRestriction クラスの詳細は、『ATG Web Commerce Programming Guide』を参照してください。

CheckOrderLimitApprovalRequirements のプロパティを、次に示します。

```
$class=atg.commerce.expression.ProcPropertyRestriction
$description=This component creates new approval rule based on Order.
$scope=global
errorMessage=Order's total is greater than the approved order limit of user.
expressionParser=/atg/commerce/util/ExpressionParser
pipelineResultErrorMessageKey=checkOrderLimitApprovalRequirements
returnValueForFalseEvaluation=1
returnValueForTrueEvaluation=0
ruleEvaluator=/atg/commerce/util/RuleEvaluator
ruleExpression=Order.priceInfo.amount > Profile.orderPriceLimit
```

ruleExpression プロパティが承認の条件を定義します。Motorprise では、このルールによって、ユーザーが作成したオーダーのオーダー合計額 Order.priceInfo.amount が、ユーザーの承認限度額 Profile.orderPriceLimit を超えているかどうかが判断されます。ユーザーが発行したオーダーがオーダー限度額を超える場合、そのオーダーは承認が必要だとマークされます。

returnValueForTrueEvaluation プロパティおよび returnValueForFalseEvaluation プロパティは、ruleExpression が true または false の場合にこのコンポーネントが返すものを定義します。後の承認パイプラインではこれらの戻り値を使用して、承認条件に基づいて適用するパイプライン・パスを決定します。

請求書支払方法における購買依頼番号をチェックする別の承認条件も作成されています。ユーザーが支払方法として請求書を選択する場合、P.O.番号または購買依頼番号を入力できます。購買依頼番号を使用する場合、ユーザーの approvalRequired プロパティが true に設定されていると、オーダーは承認が必要な状態で発行されます。詳細は、「オーダーの処理」の「*支払情報*」の項を参照してください。

Motorprise のこの新しい承認条件は、新規パイプライン・プロセッサ atg.projects.b2bstore.approval.ProcCheckRequisitionNumbers に基づく新規コンポーネント /atg/commerce/approval/processor/CheckRequisitionNumbers を作成することで実装されています。このパイプライン・プロセッサは、関連する購買依頼番号の支払グループがオーダーに含まれているかどうかに応じて、2 つの値のうちのどちらかを返します。詳細は、『ATG Web Commerce Platform API Reference』のこのクラスに関する説明を参照してください。

CheckRequisitionNumbers のプロパティを、次に示します。

```
$class=atg.projects.b2bstore.approval.ProcCheckRequisitionNumbers

# If requisitions were used, add an error to the pipeline result and
# return STOP_CHAIN_EXECUTION_AND_COMMIT so the approval pipeline will
# mark the order as requiring approval

requisitionUsedValue=0
requisitionUsedAddsPipelineError=true
requisitionUsedPipelineMessage=Requisition number found - order requires approval

# Otherwise just return a value we can use in a transition link to
# proceed to the next test for possible approval conditions.

requisitionNotUsedValue=1
```

承認条件がある場合は、それを、オーダーが発行されて精算されるときに起動される approveOrder パイプライン・チェーンに含める必要があります。承認パイプライン・フレームワークの大部分は Oracle Commerce Core Platform のレイヤーで開発および構成されており、新規のカスタム承認条件を承認パイプライン・フレームワークに追加する必要があるだけです。Core Commerce では、オーダー確認フェーズで approveOrder パイプライン・チェーンが起動され、これが次に checkRequiresApproval チェーンを起動してオーダーの承認条件を確認します。次のコードは、<ATG11dir>/B2BCommerce/config/atg/commerce/approval/approvalPipeline.xml 内の関連する承認パイプライン・コードです。

```
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->
<!-- PipelineChain used to determine if a users order requires    -->
<!-- approval or not.                                             -->
<!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ -->
<pipelinechain name="checkRequiresApproval" transaction="TX_REQUIRES_NEW"
   headlink="checkProfileApprovalRequirements">
   <pipelinelink name="checkProfileApprovalRequirements"
      transaction="TX_MANDATORY">
      <processor jndi="/atg/commerce/approval/processor/
         CheckProfileApprovalRequirements"/>
    </pipelinelink>
</pipelinechain>
```

前述のように、checkRequiredApproval チェーンによって、オーダーに承認が必要かどうかが決定されます。Motorprise レイヤー内でこのチェーンは上書きされ、次に示す<ATG11dir>/Motorprise/config/atg/commerce/approval/approvalPipeline.xml 内の新規承認条件を含みます。

```
<pipelinechain name="checkRequiresApproval"
               transaction="TX_REQUIRES_NEW"
               headlink="checkProfileApprovalRequirements"
               xml-combine="replace">
  <pipelinelink name="checkProfileApprovalRequirements"
                transaction="TX_MANDATORY">
    <processor
    jndi="/atg/commerce/approval/processor/CheckProfileApprovalRequirements"/>
   <transition returnvalue="1"
               link="checkOrderLimitApprovalRequirements"/>
  </pipelinelink>
  <pipelinelink name="checkOrderLimitApprovalRequirements"
                transaction="TX_MANDATORY">
    <processor
    jndi="/atg/commerce/approval/processor/CheckOrderLimitApprovalRequirements"/>
  </pipelinelink>
</pipelinechain>
```

```
<pipelinechain name="checkRequiresApproval" transaction="TX_REQUIRES_NEW"
   headlink="checkProfileApprovalRequirements" xml-combine="replace">
      <pipelinelink name="checkProfileApprovalRequirements" transaction=
        "TX_MANDATORY">
      <processor jndi="/atg/commerce/approval/processor/
         CheckProfileApprovalRequirements"/>
           <transition returnvalue="1" link="checkRequisitionNumbers"/>
      </pipelinelink>

      <pipelinelink name="checkRequisitionNumbers" transaction="TX_MANDATORY">
         <processor jndi="/atg/commerce/approval/processor/
            CheckRequisitionNumbers"/>
         <transition returnvalue="1" link="checkOrderLimitApprovalRequirements"/>
      </pipelinelink>

      <pipelinelink name="checkOrderLimitApprovalRequirements" transaction=
         "TX_MANDATORY">
         <processor jndi="/atg/commerce/approval/processor/
            CheckOrderLimitApprovalRequirements"/>
      </pipelinelink>
</pipelinechain>
```

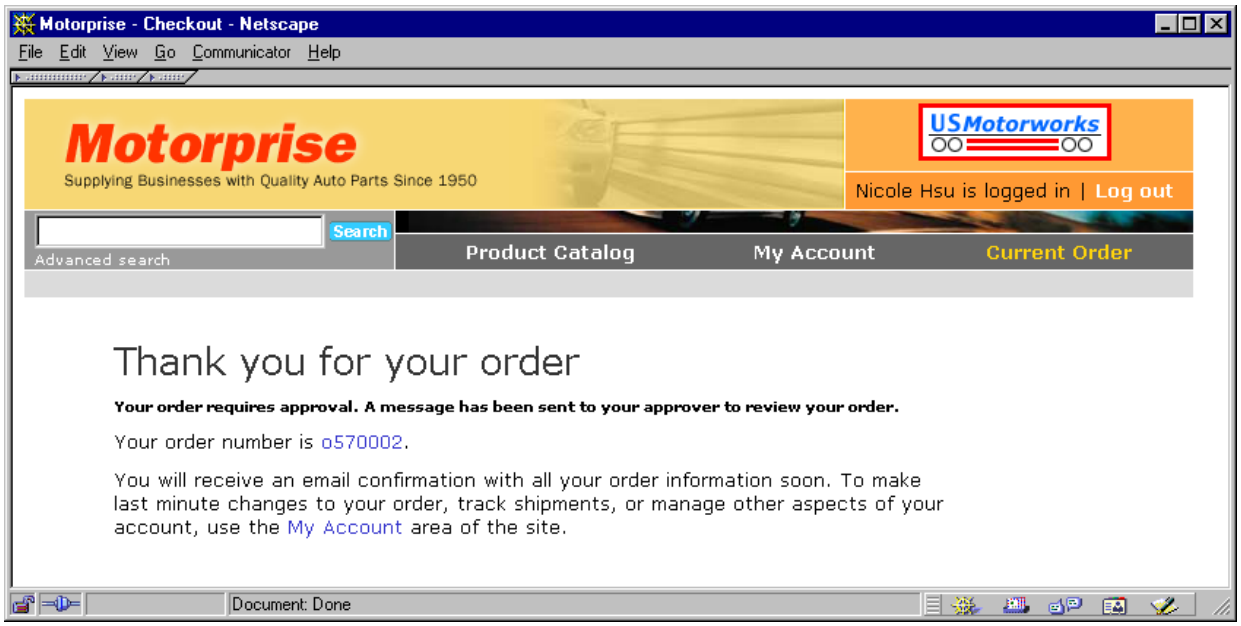
approvalPipeline.xml 内の最初のチェーンである checkRequiresApproval は精算プロセス時に起動され、新規承認条件の checkRequisitionNumbers および checkOrderLimitApprovalRequirements がこのチェーン内で起動されます。最初に、チェーンではユーザーのオーダーに承認が必要かどうかをチェックし、必要な場合は checkRequisitionNumbers 条件がチェックされます。オーダーの請求書支払方法で購買依頼番号が使用されている場合はパイプラインが停止し、オーダーは承認が必要とマークされます。オーダーが購買依頼番号を使用していない場合は checkOrderLimitApprovalRequirements に制御が移り、オーダー価格がユーザーの承認限度額を超

えているかどうかがチェックされます。オーダーに承認が必要な場合は、オーダー確認ページ thank_you.jsp に、オーダーには承認が必要という内容のメッセージが表示されます。

次のコードは、オーダーに承認が必要かどうかを判断する thank_you.jsp 内のコードです。

```
<dsp:droplet name="Switch">
  <dsp:param bean="ShoppingCart.last.state" name="value"/>
  <dsp:oparam name="5000">
    <tr>
      <td><span class=smallb>Your order requires approval. A message has been sent
          to your approver to review your order.</span><p></td>
    </tr>
  </dsp:oparam>
  <dsp:oparam name="default">
  </dsp:oparam>
</dsp:droplet>
```

ユーザーがオーダーを発行するとそれが ShoppingCart.last に移動し、その state プロパティがチェックされて、前述のように承認が必要かどうかが判断されます。



オーダーには承認が必要であることを伝えるメッセージがオーダー確認ページ(thank_you.jsp)に表示される。

請求書支払方法における購買依頼番号をチェックする別の承認条件も作成されています。ユーザーが支払方法として請求書を選択する場合、PO 番号または購買依頼番号を入力できます。購買依頼番号を選択する場合、ユーザーの approvalRequired プロパティが true に設定されていると、オーダーは承認が必要な状態で発行されます。詳細は、「オーダーの処理」の「*支払情報*」の項を参照してください。

承認者情報

承認が必要なオーダーが発行されると、そのバイヤーのオーダーを承認する担当者に E メールが送信されます。また、承認者の承認ページ（user/approvals.jsp）にオーダーが表示されます。そのリスト内のオーダーをクリックすると、オーダーを否認または承認できる画面が表示されます。

承認者は、Motorprise ホーム・ページと、「My Account」セクションから「Approval Requests」リンク（user/approvals.jsp）をクリックして、承認するオーダーを表示できます。次のコードは、保留中の承認要求を表示する approvals.jsp 内の関連 JSP コードです。

```
<%/* Pass profile id and state values to ApprovalRequiredDroplet to get all
   the orders*/%>

<dsp:droplet name="ApprovalRequiredDroplet">
   <dsp:param bean="Profile.id" name="approverid"/>
   <dsp:param name="state" value="open"/>
   <dsp:param name="startIndex" param="startIndex"/>
   <dsp:oparam name="output">

      <tr valign=top>
         <td width=33%><span class="small"><b>Order #</b></span></td>
         <td width=33%><span class="small"><b>Date ordered</b></span></td>
         <td width=33%><span class="small"><b>Status</b></span></td>
      </tr>
      <tr>
         <td colspan=3><hr size=1 color="#666666"></td>
      </tr>
      <%/* Iterate through each order and display order info */%>

      <dsp:droplet name="ForEach">
         <dsp:param name="array" param="result"/>
         <dsp:oparam name="output">
      <tr>
         <td><dsp:a href="order_pending_approval.jsp">
            <dsp:param name="orderId" param="element.id"/>
            <dsp:valueof param="element.id"/></dsp:a>
         </td>
         <td><dsp:valueof date="MMMMM d, yyyy"
            param="element.submittedDate"/></td>
         <td><dsp:valueof param="element.state"/></td>
      </tr>
   </dsp:oparam>
   <dsp:oparam name="outputEnd">
```

特定の approverid の承認が保留されているオーダーを取得するためには、ApprovalRequiredDroplet が使用されています。保留中のオーダーが必要なため、値 open を state パラメータに渡してオーダーをフェッチします。ApprovalRequiredDroplet が result パラメータで返すオーダーの配列を ForEach ドロップレットを使用して反復して、個々のオーダーを取得し、各オーダーにアクセスするリンクを提供します。

承認者の「My Account」セクションに承認が必要なオーダーが表示される。

承認ページで承認者がオーダー番号をクリックすると order_pending_approval.jsp が表示され、そこにオーダー情報とそのオーダーを承認または否認するオプションが表示されます。

「Approve Order」リンクを選択すると approve_order.jsp に移動し、オーダーに関してユーザーに表示されるメッセージをオプションで入力でき、オーダーを承認して発行し、履行できます。次のコードは、approve_order.jsp 内の関連 JSP コードです。

```
<%/* Set the order id of of approvalformahandler to the order to be approved and
   provide submit buttons for approve/cancel. */%>
<dsp:form action="approvals.jsp" method="post">
   <dsp:input bean="ApprovalFormHandler.orderId" paramvalue="orderId"
      type="hidden"/>
   <span class=help>Enter a message.</span><br>
   <dsp:textarea bean="ApprovalFormHandler.approverMessage" rows="7"
      cols="50"></dsp:textarea><p>
   <%/* page to be redirected if the order approval is successful */%>
   <dsp:input bean="ApprovalFormHandler.approveOrderSuccessURL" type="hidden"
      value="approve_confirm.jsp"/>
   <%/* page to be redirected if the order approval fails */%>
   <dsp:input bean="ApprovalFormHandler.approveOrderErrorURL" type="hidden"
      value="approve_order.jsp"/>
   <dsp:input bean="ApprovalFormHandler.approveOrder" type="submit" value="Approve
      order"/>  
   <input type="submit" value="Cancel">  
</dsp:form>
```

「Reject Order」リンクをクリックすると reject_order.jsp に移動し、ユーザーへのメッセージをオプションで入力し、オーダーを否認します。次のコードは、reject_order.jsp 内の関連 JSP コードです。

```
<dsp:form action="order_pending_approval.jsp" method="post">
   <dsp:input bean="ApprovalFormHandler.orderId" paramvalue="orderId"
      type="hidden"/>
   <span class=help>Enter a message.</span><br>
   <dsp:textarea bean="ApprovalFormHandler.approverMessage" rows="7"
      cols="50"></dsp:textarea><p>
   <input name="orderId" type="hidden" value="<dsp:valueof param="orderId"/>">
   <dsp:input bean="ApprovalFormHandler.rejectOrderSuccessURL" type="hidden"
      value="reject_confirm.jsp"/>
   <dsp:input bean="ApprovalFormHandler.rejectOrderErrorURL" type="hidden"
      value="reject_order.jsp"/>
   <dsp:input bean="ApprovalFormHandler.rejectOrder" type="submit" value=
      "Reject order"/>  
   <input type="submit" value="Cancel">  
</dsp:form>
```

承認者がオーダーを承認または否認すると、バイヤーに E メールが送信され、そのオーダーのステータスが更新されます。バイヤーはオーダーのステータスを「My Account」セクションの「Open Orders」ページ（en/user/orders_open.jsp）でも確認できます。

オーダーが承認または否認されると、メッセージの approvalStatus プロパティが approved または rejected に設定されて、Approval Update イベントが発生します。オーダーが承認または否認されたときに E メールを送信するために作成されたシナリオ ApprovalNotification では、approvalStatus に応じてこのイベントをリスニングし、承認または否認の E メールを顧客に送信します（ApprovalNotification の詳細は、このマニュアルの「マーチャンダイジング」を参照してください）。

オーダーが承認されると、オーダーは自動的に発行され、バイヤーのオーダー履歴にはオーダーのステータスが発行済として表示されます。承認者の「My Account」セクションでは、オーダーが承認済オーダー・ビューに移動します。オーダーが否認されると、オーダーは発行されず、バイヤーのオーダー履歴にはステータスが否認済として表示されます。

同様の承認条件を作成し、それを checkRequiresApproval パイプライン・チェーンに同じ方法で組み込むことができます。

承認プロセスの詳細は、『ATG Web Commerce Programming Guide』を参照してください。

7　製品カタログの表示およびアクセス

この章では、カタログ・ページを顧客に表示する方法と、顧客がカタログをナビゲートする方法について説明します。

製品テンプレート・ページ
カタログに製品および SKU を表示する製品テンプレート・ページを作成する方法について説明します。詳細なコード例が示されており、Motorprise における実装方法を理解する上で役立ちます。

カテゴリ・テンプレート・ページ
製品カタログにカテゴリを表示するカテゴリ・テンプレート・ページを作成する方法について説明します。

検索
ページに検索機能を追加する方法について説明します。

比較
ページに比較機能を追加する方法について説明します。

製品テンプレート・ページ

ユーザーが製品カタログを参照できるためには、販売する様々なカテゴリおよび製品に関する情報を表示するページを作成する必要があります。すべての製品について 1 つずつ個別の JSP を作成できますが、これは効率的ではありません。カタログに新規製品を作成したり製品を削除するたびに、変更を行う必要があります。

カタログは製品タイプ別に編成されています。ユーザーは様々なカテゴリをドリルダウンすることによって、カタログをナビゲートできます。Motorprise では、製品カテゴリおよび製品用の汎用テンプレート・ページが作成されています。これらのテンプレートは、1 つのカテゴリまたは 1 つの製品に関する一般的な情報を表示する JSP です。

テンプレートは汎用的およびデータドリブンであり、特定の製品またはカテゴリ用にハードコード化されていません。表示するすべての情報は、リポジトリから取得されます。このような汎用的で動的なページをいったん作成すると、製品カタログ内の任意のカテゴリまたは製品を表示できるため、便利です。たとえば、非常に単純なサイトの場合は、1 つのカテゴリ・テンプレート・ページ・ファイルと、1 つの製品テンプレート・ページ・ファイルの可能性があります。これらのページは動的であり、簡単に保守できます。リポジトリに製品を追加しても、カタログ・テンプレート・ページを変更する必要はありません。

Motorprise では、製品を表示する汎用製品テンプレート・ページ（`MotorpriseJSP/j2ee-apps/Motorprise/web-app/en/catalog/product.jsp`）が作成されています。これは製品の ID を受け取り、異なるコンポーネントおよびページ・テンプレートを使用してそのすべての関連情報を表示します。製品ごとに次の情報が有効です。

- 部品名

- 部品番号
- 製造業者
- 摘要
- 単位(UOM)
- 定価
- 動的価格(このユーザーに定価が適用されない場合)
- 品目の画像(一部の製品にのみ含まれる)
- 購買数量に対する割引
- 在庫状況(および企業顧客の場合は在庫レベル)

ProductLookup の使用

リポジトリから製品の属性をフェッチするためには、atg.commerce.catalog.custom.CatalogItemLookupDroplet クラスのインスタンスである /atg/commerce/catalog/ProductLookup コンポーネントが使用されています。これは、ProductLookup コンポーネントが使用された product.jsp の冒頭のコードです。

```
<dsp:droplet name="/atg/commerce/catalog/ProductLookup">
  <dsp:oparam name="output">
…
<table border=0 cellpadding=4>
<tr>
   <td>
      <span class=categoryhead>
      <dsp:valueof param="element.displayName">No name</dsp:valueof></span>
      <br>
      <b><dsp:valueof param="element.description"/></b></td>
   </td>
</tr>
<tr valign=top>
   <td>
      <dsp:include page="../common/FormError.jsp" flush="false"></dsp:include>
      <dsp:droplet name="IsEmpty">
         <dsp:param name="value" param="element.largeImage.url"/>
         <dsp:oparam name="false">
      <dsp:getvalueof id="imageURL" param="element.largeImage.url"
         idtype="java.lang.String">
         <dsp:img hspace="70" alt="Product image" src="<%=imageURL%>"/>
      </dsp:getvalueof>
      </dsp:oparam>
      </dsp:droplet>

      <dsp:getvalueof id="pval0" param="element"><dsp:include
         page="SKUProperties.jsp" flush="false"><dsp:param name="product"
         value="<%=pval0%>"/></dsp:include></dsp:getvalueof>
      <br>
      <span class=smallb>Product Description</span><br>
```

```
      <span class=small><dsp:valueof param="element.longDescription">
         No description</dsp:valueof>
      </span>
   </td>
</tr>
</table>
…
</dsp:oparam>
</dsp:droplet>
```

ProductLookup は入力として id パラメータを受け取ります。この例ではこれは製品のリポジトリ ID で、ユーザーがカテゴリ・ページでクリックした製品リンクから取得されます。次に、element パラメータが、渡された id の製品にバインドされます。このコード例では、ProductLookup コンポーネントに id パラメータが明示的に渡されていません。ページの先頭でページレベルの id パラメータが入力パラメータとしてすでに定義されており、これが ProductLookup に暗黙に渡されます。

次のコードは、製品の longDescription プロパティを表示する方法、または製品が longDescription 値を持たない場合はデフォルト値を表示する方法を示しています。

```
<dsp:valueof param="element.longDescription">No description</dsp:valueof>
```

次のコード例は、製品の画像が null でない場合にこれを表示する方法を示しています。

```
<dsp:droplet name="IsEmpty">
   <dsp:param name="value" param="element.largeImage.url"/>
   <dsp:oparam name="false">
      <dsp:getvalueof id="imageURL" param="element.largeImage.url"
         idtype="java.lang.String">
      <dsp:img hspace="70" alt="Product image" src="<%=imageURL%>"/>
      </dsp:getvalueof>
   </dsp:oparam>
</dsp:droplet>
```

SKU 情報

Motorprise カタログには、製品ごとに 1 つの SKU があります。各製品の SKU は、product をパラメータとして渡して product.jsp から起動される、SKUProperties.jsp を使用して表示されます。部品番号、製造業者、可用性のステータス、価格など、各 SKU の属性が表示されます。次の JSP コードを使用して、この情報が表示されます。

```
<dsp:droplet name="ForEach">
  <dsp:param name="array" param="product.childSKUs"/>
  <dsp:oparam name="output">
    <table cellspacing="0" cellpadding="0" border="0">
      <tr>
        <td> </td>
        <td><span class=smallb>Part Number:</span></td>
        <td> </td>
        <td><span class=small><dsp:valueof param=
           "element.manufacturer_part_number">No id</dsp:valueof></span>
```

```
      <td> </td>
    </tr>
    <tr>
      <td> </td>
      <td><span class=smallb>Manufacturer:</span></td>
      <td> </td>
      <td><span class=small><dsp:valueof param=
         "product.manufacturer.displayName">Unknown</dsp:valueof>
<td> </td>
    </tr>

    <dsp:droplet name="/atg/dynamo/droplet/Switch">
      <dsp:param bean="Profile.transient" name="value"/>
      <dsp:oparam name="false">
        <tr>
          <td> </td>
          <td><span class=smallb>Availability:</span></td>
          <td> </td>
          <dsp:droplet name="/atg/commerce/inventory/InventoryLookup">
            <dsp:param name="itemId" param="element.repositoryId"/>
            <dsp:oparam name="output">
              <td><span class=small><dsp:valueof param=
                "inventoryInfo.availabilityStatusMsg">
                Unknown</dsp:valueof></span>
            </dsp:oparam>
          </dsp:droplet>
          <td> </td>
        </tr>

        <dsp:droplet name="/atg/dynamo/droplet/Switch">
          <dsp:param bean="Profile.parentOrganization.customerType"
             name="value"/>
           <dsp:oparam name="Enterprise">
             <tr>
               <td> </td>
               <td><span class=smallb>Stock Level:</span></td>
               <td> </td>
               <dsp:droplet name="/atg/commerce/inventory/InventoryLookup">
                 <dsp:param name="itemId" param="element.repositoryId"/>
                 <dsp:oparam name="output">
                   <td><span class=small><dsp:valueof
                      param="inventoryInfo.stockLevel">
                      Unknown</dsp:valueof></span>
                 </dsp:oparam>
               </dsp:droplet>
               <td> </td>
             </tr>
           </dsp:oparam>
         </dsp:droplet>
```

```
          <dsp:getvalueof id="pval0" param="product"><dsp:getvalueof id="pval1"
             param="element"><dsp:include page="DisplayPrice.jsp"
             flush="true"><dsp:param name="Product"
             value="<%=pval0%>"/><dsp:param name="Sku"
             value="<%=pval1%>"/></dsp:include></dsp:getvalueof></dsp:getvalueof>
        </dsp:oparam>
      </dsp:droplet>
    </table>
  </dsp:oparam>
</dsp:droplet>
```

在庫情報

すべての登録ユーザーに製品の可用性ステータス(在庫あり、または在庫切れなど)を表示するためには、InventoryLookup が使用されています。さらに、バイヤーが USMW などの企業顧客に所属する場合は、実際の在庫量も表示されます。匿名ユーザーまたはゲスト・ユーザーには、在庫情報や価格は表示されません。catalog/SKUProperties.jsp からの次のコードを使用して、この情報が表示されます。

```
<dsp:droplet name="/atg/dynamo/droplet/Switch">
        <dsp:param bean="Profile.transient" name="value"/>
        <dsp:oparam name="false">
          <tr>
            <td> </td>
            <td><span class=smallb>Availability:</span></td>
            <td> </td>
            <dsp:droplet name="/atg/commerce/inventory/InventoryLookup">
              <dsp:param name="itemId" param="element.repositoryId"/>
              <dsp:oparam name="output">
                <td><span class=small><dsp:valueof param=
                  "inventoryInfo.availabilityStatusMsg">
                  Unknown</dsp:valueof></span>
              </dsp:oparam>
            </dsp:droplet>
            <td> </td>
          </tr>

          <dsp:droplet name="/atg/dynamo/droplet/Switch">
            <dsp:param bean="Profile.parentOrganization.customerType"
              name="value"/>
             <dsp:oparam name="Enterprise">
               <tr>
                 <td> </td>
                 <td><span class=smallb>Stock Level:</span></td>
                 <td> </td>
                 <dsp:droplet name="/atg/commerce/inventory/InventoryLookup">
                   <dsp:param name="itemId" param="element.repositoryId"/>
                   <dsp:oparam name="output">
                     <td><span class=small><dsp:valueof param=
```

```
                    "inventoryInfo.stockLevel">Unknown</dsp:valueof></span>
                </dsp:oparam>
              </dsp:droplet>
              <td> </td>
            </tr>
          </dsp:oparam>
        </dsp:droplet>
        <dsp:getvalueof id="pval0" param="product"><dsp:getvalueof id="pval1"
          param="element"><dsp:include page="DisplayPrice.jsp"
          flush="true"><dsp:param name="Product" value="<%=pval0%>"/><dsp:param
          name="Sku" value="<%=pval1%>"/>
          </dsp:include></dsp:getvalueof></dsp:getvalueof>
      </dsp:oparam>
    </dsp:droplet>
```

製品価格の表示

Motorprise では価格表を使用して異なる製品および SKU の価格を管理しています。価格表を使用することで、特定の一連の価格を特定の顧客グループに対してターゲット設定できます。Motorprise では、定価と数量価格の 2 種類の価格設定が使用されています。定価は、割引が適用されない、製品の正規の値段です。数量価格設定では、オーダーされる品目数に基づいて価格が決定されます。Motorprise では、2 種類の複合的な数量価格設定が使用されます。

- **一括価格設定:** 最低オーダー量に基づいて製品の価格が計算されます。
- **段階的価格設定:** 異なる価格設定レベルの定量または固定重量を使用して製品の価格が計算されます。

次の 2 つのページ・フラグメントを使用して、製品テンプレート・ページに価格を表示しています: `MotorpriseJSP/j2ee-apps/Motorprise/web-app/en/catalog/DisplayPrice.jsp` および `MotorpriseJSP/j2ee-apps/Motorprise/web-app/en/common/DisplayComplexPrice.jsp`。`DisplayPrice.jsp` は、定価を書式設定して表示します。`DisplayComplexPrice.jsp` は、価格レベル数、各レベルの最小価格と最大価格など、複合価格の詳細な情報を表示します。

`DisplayPrice.jsp` が起動されて、製品の価格を表示します。特定の製品および SKU で使用される価格設定方式を決定するためには、`PriceDroplet` が使用されています。これは SKU および製品リポジトリ項目を入力として受け取り、価格リポジトリ項目を出力として提供します。`price` パラメータには、使用される価格設定方式、価格表、複合価格など、すべての価格設定属性が含まれています。製品の価格を表示するためは、次の JSP フラグメントが使用されています。

```
<dsp:droplet name="PriceDroplet">
  <dsp:param name="product" param="Product"/>
  <dsp:param name="sku" param="Sku"/>
  <dsp:oparam name="output">
    <dsp:droplet name="Switch">
      <dsp:param name="value" param="price.pricingScheme"/>
      <dsp:oparam name="listPrice">
        <tr>
          <td> </td>
          <td><span class=smallb>Price:</span></td>
```

```
<td>  </td>
<td>
```

定価をそのロケールにあわせて適切に書式設定するためには、CurrencyConversionFormatter ドロップレットが使用されています。locale（price.listPrice が定義されているロケール）と targetLocale（価格を表示する対象のロケール）が同じであることに注意してください。これは、実際には通貨の変換は行われず、書式設定のみが行われることを意味します。結果は Profile.pricelist.locale にあわせて適切に書式設定されます。デフォルト通貨がユーロの場合は、euroSymbol プロパティを設定することにより正しい通貨記号を使用します。

```
<%/*Display price for locale of user: */%>
<dsp:droplet name="CurrencyConversionFormatter">
     <dsp:param name="currency" param="price.listPrice"/>
     <dsp:param bean="Profile.priceList.locale" name="locale"/>
     <dsp:param bean="Profile.priceList.locale" name="targetLocale"/>
     <dsp:param name="euroSymbol" value="&euro;"/>
     <dsp:oparam name="output">
       <span class=small><dsp:valueof valueishtml="<%=true%>"
          param="formattedCurrency">no price</dsp:valueof></span>
     </dsp:oparam>
   </dsp:droplet>

<%/*Switch to see whether user is of German locale */%>
<dsp:droplet name="Switch">
         <dsp:param bean="Profile.priceList.locale" name="value"/>
         <dsp:oparam name="de_DE_EURO">

         <%/*Display price in DM for German users: */%>
```

この時点では以前と同じ価格が表示されており、ドイツのユーザー向けにこれを正しく表示する必要があります。ここでは、locale と targetLocale は同じではありません。つまり、ターゲット・ロケール用に結果を書式設定する前に、通貨（price.listPrice）を de_DE_EURO のデフォルト通貨から de_DE のデフォルト通貨に変換する必要があります。/atg/droplet/ExchangeRates.properties に格納されている為替レートに基づいて値が計算されます。その後で、de_DE ロケール用に結果が書式設定されます（このようにして、ユーロで格納されていた価格がドイツ・マルクで表示されます）。

```
         <dsp:droplet name="CurrencyConversionFormatter">
           <dsp:param name="currency" param="price.listPrice"/>
           <dsp:param bean="Profile.priceList.locale" name="locale"/>
           <dsp:param name="targetLocale" value="de_DE"/>
           <dsp:oparam name="output">
             <span class=small>(<dsp:valueof
                param="formattedCurrency">no price
                </dsp:valueof>)</span>
           </dsp:oparam>
         </dsp:droplet>
         </dsp:oparam>
</dsp:droplet>
```

```
          </td>
          <td> </td>
        </tr>
      </dsp:oparam>

      <dsp:oparam name="bulkPrice">
        <dsp:getvalueof id="pval0" param="price.complexPrice"><dsp:getvalueof
          id="pval1" param="price.pricingScheme"><dsp:include page=
          "DisplayComplexPrice.jsp" flush="true"><dsp:param name="complexPrice"
          value="<%=pval0%>"/><dsp:param name="pricingScheme"
          value="<%=pval1%>"/></dsp:include></dsp:getvalueof></dsp:getvalueof>
      </dsp:oparam>

      <dsp:oparam name="tieredPrice">
        <dsp:getvalueof id="pval0" param="price.complexPrice"><dsp:getvalueof
           id="pval1" param="price.pricingScheme"><dsp:include
           page="DisplayComplexPrice.jsp" flush="true"><dsp:param name=
           "complexPrice" value="<%=pval0%>"/><dsp:param name="pricingScheme"
           value="<%=pval1%>"/></dsp:include></dsp:getvalueof></dsp:getvalueof>
      </dsp:oparam>
    </dsp:droplet>

  </dsp:oparam>
  <dsp:oparam name="error">
    There was a pricing error.
  </dsp:oparam>

</dsp:droplet>
```

前述のコードでは、価格設定方式が定価の場合は、その値が書式設定されて表示されます。それ以外の場合は、DisplayComplexPrice.jsp が起動されて複合価格が表示されます。

価格レベル数ならびに各レベルの最小価格および最大価格など、複合価格の詳細な情報を表示するためには、ComplexPriceDroplet が使用されています。次の JSP コード・フラグメントは、複合価格の各レベルを反復して、その価格設定情報を表示する方法を示しています。

```
      <tr>
        <td> </td>
        <td valign=top><span class=smallb>Bulk Price:</span></td>
        <td> </td>
        <td>
          <table border=0 cellspacing=0 cellpadding=0 width=100%>

      <dsp:droplet name="ComplexPriceDroplet">
        <dsp:param name="complexPrice" param="complexPrice"/>
        <dsp:oparam name="output">
          <dsp:droplet name="For">
            <dsp:param name="howMany" param="numLevels"/>
            <dsp:param name="indexName" value="index"/>
```

```
<dsp:oparam name="output">
  <tr>
    <td><span class=small>
        <dsp:valueof param="levelMinimums[param:index]"/>
    </span></td>
    <td>
         - 
    </td>
    <td><span class=small>
        <dsp:valueof param="levelMaximums[param:index]">or
           over</dsp:valueof>
    </span></td>
    <td>

      <%/*Display price for locale of user: */%>
      <dsp:droplet name="CurrencyConversionFormatter">
          <dsp:param name="currency" param="prices[param:index]"/>
          <dsp:param bean="/atg/userprofiling/
             Profile.priceList.locale" name="locale"/>
          <dsp:param bean="/atg/userprofiling/
             Profile.priceList.locale" name="targetLocale"/>
          <dsp:param name="euroSymbol" value="&euro;"/>
          <dsp:oparam name="output">
              <span class=small><dsp:valueof
               valueishtml="<%=true%>" param="formattedCurrency">no
                  price</dsp:valueof></span>
          </dsp:oparam>
        </dsp:droplet>
        <dsp:droplet name="Switch">
          <dsp:param bean="/atg/userprofiling/
             Profile.priceList.locale" name="locale"/>
          <dsp:oparam name="default">
          </dsp:oparam>
          <dsp:oparam name="de_DE">

        <%/*Don't Display in en directory */%>
        <%/*
        <dsp:droplet name="CurrencyConversionFormatter">
          <dsp:param name="currency" param="prices[param:index]"/>
          <dsp:param bean="/atg/userprofiling/
             Profile.priceList.locale" name="locale"/>
          <dsp:param name="targetLocale" value="de_DE"/>
          <dsp:oparam name="output">
            <span class=small>(<dsp:valueof param=
               "formattedCurrency">no price</dsp:valueof>)</span>
          </dsp:oparam>
        </dsp:droplet> */%>
        </dsp:oparam>
        </dsp:droplet>
```

```
                </td>
              </tr>
            </dsp:oparam>
          </dsp:droplet>
        </dsp:oparam>
      </dsp:droplet>
    </table>
    </td>
  </tr>
</dsp:oparam><!--  ************************************ End of Bulk Pricing -->
```

段階的価格を表示するために、同じ方法が使用されています。

通貨の書式設定

前述のコード例では、渡された targetLocale パラメータに従って通貨を書式設定するために、CurrencyConversionFormatterドロップレットが使用されています。入力として書式設定する通貨、つまりユーザーのロケール targetLocale を受け取り、formattedCurrency パラメータを出力します。

オーダーへの追加

顧客に対して製品が表示されると、登録顧客の場合は、製品ページ上でその品目の数量を現在のオーダーに追加できます。

次のコードに示すように、現在のオーダーに製品を追加するためには、CartModifierFormHandler が使用されています。product パラメータが AddToCart.jsp に渡されて、製品の追加に使用されます。

```
<input name="id" type="hidden" value="<dsp:valueof param=
   "product.repositoryId"/>">
   <dsp:input bean="CartModifierFormHandler.addItemToOrderSuccessURL"
      type="hidden" value="../checkout/cart.jsp?noCrumbs=false"/>
   <dsp:input bean="CartModifierFormHandler.SessionExpirationURL" type="hidden"
      value="../common/session_expired.jsp"/>
   <dsp:input bean="CartModifierFormHandler.productId"
      paramvalue="product.repositoryId" type="hidden"/>
   <dsp:droplet name="/atg/dynamo/droplet/ForEach">
      <dsp:param name="array" param="product.childSKUs"/>
      <dsp:oparam name="output">
      <table border=0 cellpadding=3 width=100%>
         <tr>
         <td>
         <dsp:input bean="CartModifierFormHandler.catalogRefIds"
            paramvalue="element.repositoryId" type="hidden"/>
         <span class=smallb>Qty</span> 
         <dsp:input bean="CartModifierFormHandler.quantity" size="4" type="text"
            value="1"/>  
         <dsp:input bean="CartModifierFormHandler.addItemToOrder" type="submit"
            value="Add to order"/>
         <br>
```

```
        </td>
        </tr>
      </table>

      </dsp:oparam>
   </dsp:droplet>
```

品目がオーダーに追加されると、「**Add to Order**」ボタンによってユーザーが/checkout/cart.jsp にリダイレクトされます。それ以外の場合は、ユーザーは同じページにリダイレクトされ、エラーが表示されます。すべての SKU が反復されて、SKU ごとにユーザーに対して入力オプションが提供され、それが CartModifierFormHandler.catalogRefIds に追加されます。

注意: Motorprise はデモ・サイトのため、各製品の SKU は 1 つのみです。しかし、製品のすべての SKU を反復するコードが含まれているため、カタログに複数の SKU からなる製品が含まれている場合でも、このコードを簡単に適用できます。

リストへの追加

顧客がその時点で製品を購入するのではなく、品目を保存して後で購入する場合、AddToList.jsp からの次のコード・フラグメントに示すように、任意の既存の購入リストに保存できます。

```
<dsp:form action="product.jsp" method="post">
   <input name="id" type="hidden" value="<dsp:valueof param=
      "product.repositoryId"/>">
   <dsp:input bean="PurchaselistFormHandler.addItemToPurchaselistErrorURL"
      type="hidden" value="product.jsp"/>
   <dsp:input bean="PurchaselistFormHandler.productId"
      paramvalue="product.repositoryId" type="hidden"/>
   <dsp:droplet name="/atg/dynamo/droplet/ForEach">
      <dsp:param name="array" param="product.childSKUs"/>
      <dsp:oparam name="output">
         <table border=0 cellpadding=3 width=100%>
            <tr>
               <td><dsp:input bean="PurchaselistFormHandler.catalogRefIds"
                  paramvalue="element.repositoryId" type="hidden"/>
               <span class=smallb>Qty</span> 
               <dsp:input bean="PurchaselistFormHandler.quantity" size="2"
                  type="text" value="1"/> 
      </dsp:oparam>
   </dsp:droplet>

   <dsp:select bean="PurchaselistFormHandler.purchaseListId">
      <dsp:droplet name="ForEach">
         <dsp:param bean="Profile.purchaselists" name="array"/>
         <dsp:oparam name="output">
            <dsp:getvalueof id="elem" idtype="atg.repository.RepositoryItem"
               param="element">
            <dsp:option value="<%=elem.getRepositoryId()%>"/>
            <dsp:valueof param="element.eventName">Unnamed Purchase List
```

```
            </dsp:valueof>
            </dsp:getvalueof>
         </dsp:oparam>
      </dsp:droplet>
   </dsp:select></td>
   </tr>
   <tr>
      <td><dsp:input bean="PurchaselistFormHandler.addItemToPurchaselist"
         type="submit" value="Add to list"/></td>
   </tr>

   <tr>
      <td>
         <table border=0 cellpadding=3 width=100%>
         <tr>
            <td><span class=smallb><dsp:a href=
               "../user/purchase_lists.jsp?noCrumbs=false"><dsp:param
               name="product" param="product.repositoryId"/><dsp:param
               name="noCrumbs" value="false"/>Create new purchase
               list</dsp:a></span></td>
         </tr>
         </table>
      </td>
   </tr>

</table>
</dsp:form>
```

特定の品目を任意の購入リストに追加するためには、PurchaseListFormHandler が使用されています。製品の SKU をすべて反復して品目の数量の入力オプションを提供し、顧客の既存の購入リストをすべてドロップダウン・ボックスに表示して、その中の 1 つを選択するようになっています。

新規購入リストの作成

購入リストに品目を追加できることに加えて、ユーザーは新規購入リストも作成できます。**「Create new purchase list」**リンクによってユーザーは「My Account」内の「Purchase Lists」ページ（user/purchase_lists.jsp）に移動し、そこで新規リスト名を入力して**「Create list」**ボタンをクリックできます。購入リストの作成の詳細は、このマニュアルの「自分のアカウント」の「*購入リストの作成*」の項を参照してください。

カテゴリ・テンプレート・ページ

製品を表示するテンプレート・ページを作成するのと同様に、カテゴリを表示するテンプレート・ページも作成します。Motorprise には 3 種類のカテゴリがあります。他の子カテゴリが含まれるルート・カテゴリ、他の子カテゴリが含まれる子カテゴリ、および製品が含まれる子カテゴリです。これらのタイプのカテゴリを表示するために、3 つの汎用テンプレート・ページが開発されました。

/en/common/CatalogNav.jsp

/en/catalog/category.jsp

/en/catalog/sub_category.jsp

CatalogNav.jsp は、ユーザーのルート・カテゴリを表示する場合に使用されます。Motorprise では、各ユーザーにカタログが割り当てられています。ユーザーのプロファイルにカタログが定義されていない場合は、parentOrganization の契約から取得されます。このような各カタログには、前述の 3 種類のカテゴリが含まれています。左ナビゲーション・ペインを提供するために、カタログ全体で CatalogNav.jsp が使用されています。

CatalogNav.jsp を使用して左ナビゲーション・ペインにカテゴリが表示される。

CatalogNav.jsp は、次に示すように、ユーザーのすべてのルート・カテゴリを反復して表示します。

```
<dsp:droplet name="/atg/dynamo/droplet/ForEach">
   <dsp:param  name="array" bean=
      "/atg/userprofiling/Profile.catalog.allRootCategories"/>
   <dsp:oparam name="output">
      <tr bgcolor="#FFFFFF">
         <td colspan=2><dsp:img src="../images/d.gif"/></td>
      </tr>
      <tr bgcolor="F7D774">
         <td><dsp:img src="../images/d.gif" hspace="5"/></td>
         <td><dsp:img src="../images/d.gif" vspace="1"/><br>

            <dsp:getvalueof id="urlStr" idtype="java.lang.String"
               param="element.template.url">
            <dsp:a page="<%=urlStr%>">
```

```
                <dsp:param name="id" param="element.repositoryId"/>
                <dsp:param name="navAction" value="pop"/>
                <dsp:param name="Item" param="element"/>
                   <b><font size=-1 color="#555555"><dsp:valueof param=
                      "element.displayName"/></font></b>
             </dsp:a>
             </dsp:getvalueof>

      <br>
             <dsp:img src="../images/d.gif" vspace="1"/><br></td>
      </tr>
   </dsp:oparam>

   <dsp:oparam name="empty">
      <p>No root categories found.
   </dsp:oparam>
</dsp:droplet>
```

カテゴリの `template.url` プロパティには、カテゴリの表示に使用するテンプレートの URL が含まれています。

`Category.jsp` は、他の子カテゴリが含まれる子カテゴリを表示する場合に使用されます。たとえば、「Exhaust」カテゴリには、「Reducers」、「Gaskets」、「Headers」、「Mufflers」および「Pipes」の各サブカテゴリがあります。

「Exhaust」カテゴリには、「Reducers」、「Gaskets」、「Headers」、「Mufflers」および「Pipes」の各サブカテゴリがある。

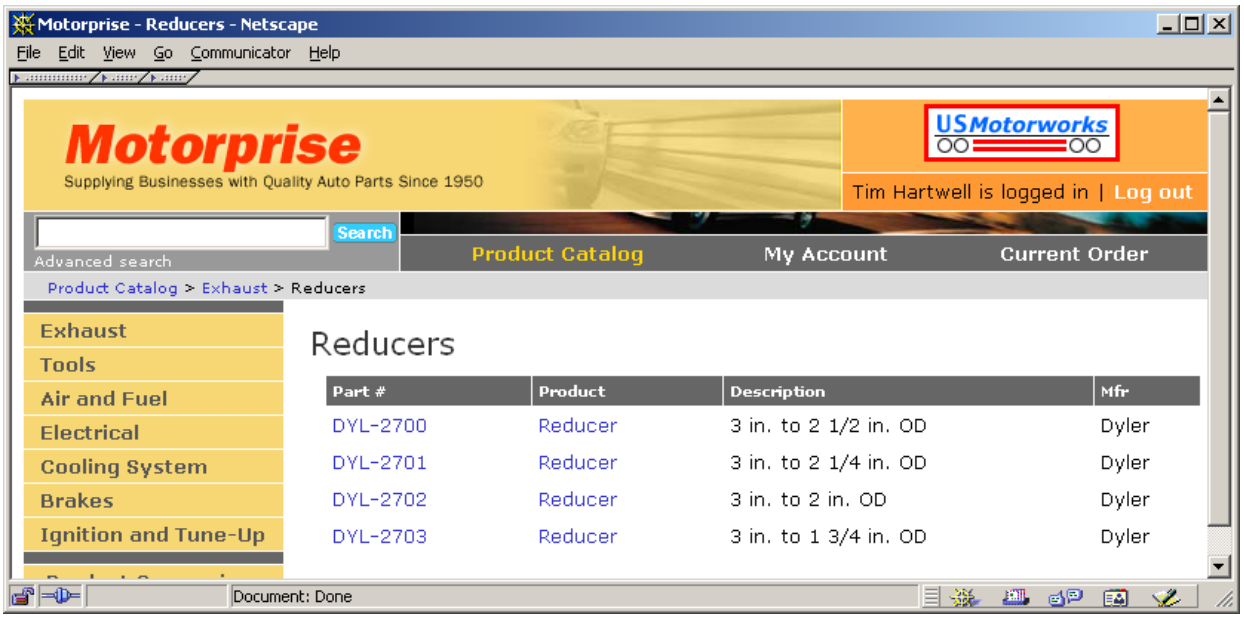
`Category.jsp` はカテゴリの `id` をページ・パラメータとして受け取り、`CategoryLookup` を使用してカテゴリを取得し、次に示すように子カテゴリを表形式で表示します。

```
<dsp:droplet name="/atg/commerce/catalog/CategoryLookup">
   <dsp:oparam name="output">

   <dsp:getvalueof id="page_title" param="element.displayName">
   <dsp:include page="../common/HeadBody.jsp" flush="true">
      <dsp:param name="pagetitle" value="<%=page_title%>"/>
   </dsp:include>
   </dsp:getvalueof>

   <table border=0 cellpadding=0 cellspacing=0 width=800>
      <tr>
         <td colspan=2><dsp:include page="../common/BrandNav.jsp"
            flush="true"></dsp:include></td>
      </tr>

      <tr bgcolor="#DBDBDB">
         <td colspan=2 height=18><span class="small">  
            <dsp:droplet name="ArrayIncludesValue">
               <dsp:param name="array" bean="Profile.catalog.allrootcategories"/>
               <dsp:param name="value" param="element"/>
               <dsp:oparam name="false">
               <dsp:include page="../common/breadcrumbs.jsp" flush="true">
                  <dsp:param name="displaybreadcrumbs"
                     value="true"/></dsp:include>
          </dsp:oparam>
          <dsp:oparam name="true">
            <dsp:include page="../common/breadcrumbs.jsp" flush="true"><dsp:param
               name="displaybreadcrumbs" value="true"/><dsp:param name="navAction"
               value="jump"/><dsp:param name="navCount" value="0"/></dsp:include>
          </dsp:oparam>
        </dsp:droplet>
       </span>
      </td>
    </tr>


    <tr valign=top>
      <td width=175>
        <!-- category navigation -->
        <dsp:include page="../common/CatalogNav.jsp" flush="true"></dsp:include>
        <!-- incentives slot -->
        <dsp:include page="../common/Incentive.jsp" flush="true"></dsp:include>

      </td>
      <td width=625>
      <!-- promotion slot -->
      <table border=0 cellpadding=4 width=100%>
        <!--this row used to ensure proper spacing of table cell-->
        <tr><td colspan=2><dsp:img src="../images/d.gif" hspace="304"/></td></tr>
```

```
        <tr>
          <td colspan=2> <span class="categoryhead">
            <dsp:valueof param="element.itemDisplayName">No name
            </dsp:valueof></span></td>
        </tr>
        <tr>
      <td>

        <dsp:droplet name="/atg/dynamo/droplet/ForEach">
          <dsp:param name="array" param="element.childCategories"/>
          <dsp:oparam name="outputStart">
            <table border=0 cellpadding=3 width=100%>
          </dsp:oparam>
          <dsp:oparam name="output">
            <tr>
              <td width=3><dsp:img src="../images/d.gif" hspace="1"/></td>
              <td>
                <dsp:getvalueof id="urlStr" idtype="java.lang.String"
                   param="element.template.url">
          <dsp:a page="<%=urlStr%>">
            <dsp:param name="id" param="element.repositoryId"/>
            <dsp:valueof param="element.displayName">No name</dsp:valueof>
          </dsp:a>
                </dsp:getvalueof>
                <br> <dsp:valueof param="element.description"/>
              </td>
            </tr>
          </dsp:oparam>
          <dsp:oparam name="empty">
            <tr>
              <td width=3></td>
              <td>No child categories found.</td>
            </tr>
          </dsp:oparam>
        </dsp:droplet>
            </table>
            </td>
          </tr>
    </table>
    </td>
  </tr>
 </table>
  </dsp:oparam>
</dsp:droplet>
```

CategoryLookup はカテゴリの id を入力として使用します。しかし、カテゴリの id パラメータが CategoryLookup に明示的に渡されることはありません。これはページレベルのパラメータとしてすでに定義されており、カテゴリ・オブジェクトを出力します。カテゴリの childCategories を反復してカテゴリとその属性を表示し、そのカテゴリを表示する URL も提供します。

sub_category.jsp は、カテゴリの id をページ・パラメータとして受け取ることによって、製品が含まれるカテゴリを表示する場合に使用されます。



CategoryLookup は、カテゴリに含まれる製品を表示する場合に使用されます。JSP フラグメントを次に示します。

```
<dsp:droplet name="/atg/commerce/catalog/CategoryLookup">
   <dsp:oparam name="output">

   <dsp:getvalueof id="page_title" param="element.displayName">
   <dsp:include page="../common/HeadBody.jsp" flush="true">
      <dsp:param name="pagetitle" value="<%=page_title%>"/>
    </dsp:include>
    </dsp:getvalueof>

    <table border=0 cellpadding=0 cellspacing=0 width=800>
    <tr>
      <td colspan=2><dsp:include page="../common/BrandNav.jsp" flush=
        "true"></dsp:include></td>
    </tr>

    <tr bgcolor="#DBDBDB">
      <td colspan=2 height=18><span class="small">  
        <dsp:droplet name="ArrayIncludesValue">
          <dsp:param name="array" bean="Profile.catalog.rootcategories"/>
          <dsp:param name="value" param="element"/>
          <dsp:oparam name="false">
            <dsp:include page="../common/breadcrumbs.jsp" flush="true"><dsp:param
              name="displaybreadcrumbs" value="true"/></dsp:include>
```

```
      </dsp:oparam>
      <dsp:oparam name="true">
        <dsp:include page="../common/breadcrumbs.jsp" flush="true"><dsp:param
          name="displaybreadcrumbs" value="true"/><dsp:param name="navAction"
          value="jump"/><dsp:param name="navCount" value="0"/></dsp:include>
      </dsp:oparam>
    </dsp:droplet>
     </span>
   </td>
 </tr>

<tr valign=top>
  <td width=175>
  <!-- left panel -->
  <dsp:include page="../common/CatalogNav.jsp" flush="true"></dsp:include>
  <!-- incentives slot -->
  <dsp:include page="../common/Incentive.jsp" flush="true"></dsp:include>
  </td>
  <td width=625><!-- main content -->

  <table border=0 cellpadding=4 width=100%>
    <!--this row used to ensure proper spacing of table cell-->
    <tr><td colspan=2><dsp:img src="../images/d.gif" hspace="304"/></td></tr>
<tr>
      <td colspan=2>
       <span class="categoryhead">
      <dsp:valueof param="element.itemDisplayName">No name
        </dsp:valueof></span>
      <br><dsp:img src="../images/d.gif" vspace="4"/><br>
        <table border=0 cellpadding=4 cellspacing=1 width=100%>
          <dsp:droplet name="/atg/dynamo/droplet/ForEach">
            <dsp:param name="array" param="element.childProducts"/>
            <dsp:oparam name="outputStart">
              <tr valign="bottom">
                <td><dsp:img src="../images/d.gif" hspace="1"/></td>
                <td bgcolor="#666666" colspan=2><span class="smallbw">Part #
                </span></td>
                <td bgcolor="#666666" colspan=2><span class="smallbw">
                  Product</span></td>
                <td bgcolor="#666666" colspan=2><span class=
                 "smallbw">Description</span></td>
                <td bgcolor="#666666" colspan=2><span class=
                  "smallbw">Mfr</span></td>
              </tr>
            </dsp:oparam>
            <dsp:oparam name="output">
              <tr valign="top">
        <td><dsp:img src="../images/d.gif" hspace="1"/></td>
                <dsp:droplet name="/atg/dynamo/droplet/ForEach">
```

```
          <dsp:param name="array" param="element.childSKUs"/>
          <dsp:param name="elementName" value="sku"/>
          <dsp:oparam name="output">
           <td><nobr>
           <dsp:getvalueof id="skuURL" idtype="java.lang.String"
             param="element.template.url">
    <dsp:a page="<%=skuURL%>">
      <dsp:param name="id" param="element.repositoryId"/>
            <dsp:valueof param="sku.manufacturer_part_number">No part
              number</dsp:valueof>
    </dsp:a>
          </nobr></td>
    </dsp:getvalueof>
         </dsp:oparam>
       </dsp:droplet>

       <td> </td>
       <td>
<dsp:getvalueof id="urlStr" idtype="java.lang.String"
  param="element.template.url">
<dsp:a page="<%=urlStr%>">
  <dsp:param name="id" param="element.repositoryId"/>
  <dsp:valueof param="element.displayName">No name</dsp:valueof>
</dsp:a>
</dsp:getvalueof>
       </td>

       <td> </td>
       <td><dsp:valueof param="element.description">No description
       </dsp:valueof></td>

       <td> </td>
       <td><dsp:valueof param=
         "element.manufacturer.displayName">Unknown
       </dsp:valueof></td>



     </tr>
  </dsp:oparam>
  <dsp:oparam name="outputEnd">
    <tr>
      <td> </td>
    </tr>
  </dsp:oparam>
  <dsp:oparam name="empty">
    <tr><td>  No child products.</td></tr>
  </dsp:oparam>
</dsp:droplet>
```

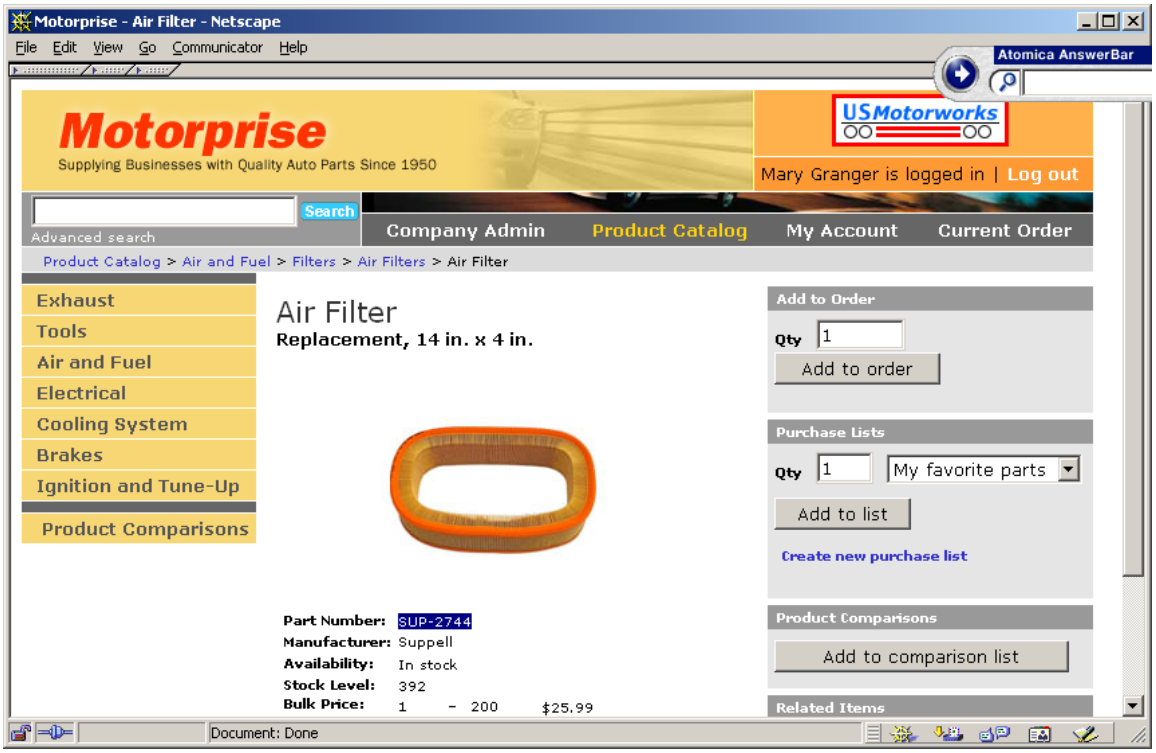
```
            </table>
          </td>
        </tr>
      </table>
      </td>
    </tr>
  </table>
  </dsp:oparam>
</dsp:droplet>
```

前述のコード・サンプルは、CategoryLookup を使用して id に対応するカテゴリを取得し、製品および各製品の SKU を反復して SKU とその属性を表示するようになっています。

履歴ナビゲーション

各テンプレート・ページの上部には、顧客が現在のページにいたるまでに訪問したカテゴリの階層をリストすることによって、履歴ナビゲーション情報が表示されます。

たとえば、顧客が Motorprise カタログの先頭から「Air and Fuel」カテゴリ、「Filters」サブカテゴリ、「Air Filters」サブカテゴリのページを経て、SUP-2744 エアー・フィルタの製品ページへとナビゲートした場合、ナビゲーション履歴（ページ上部）は次のようになります。



階層ナビゲーション内の各項目はリンクになっています。たとえば、ユーザーが「Air and Fuel」をクリックすると、「Air and Fuel」のカテゴリ・テンプレート・ページに移動します。

顧客がホーム・ページから SUP-2744 エアー・フィルタの製品ページに直接移動した場合も（たとえば、ホーム・ページ上の特集製品から）、履歴ナビゲーションにはカタログの階層構造が引き続き反映されます。ナビゲーション・パスは顧客にとって直感的であることが望ましいため、この品目がカタログのどこにあるのかをユーザーが把握できるように、履歴ナビゲーション・リストはカテゴリ・ページを反映しています。これを行うには、jump に設定された navAction パラメータを使用して、CatalogNavHistoryCollector コンポーネントを起動します。

履歴ナビゲーションの使用には、情報の収集および情報の表示という 2 つの要素があります。

ナビゲーション履歴の収集

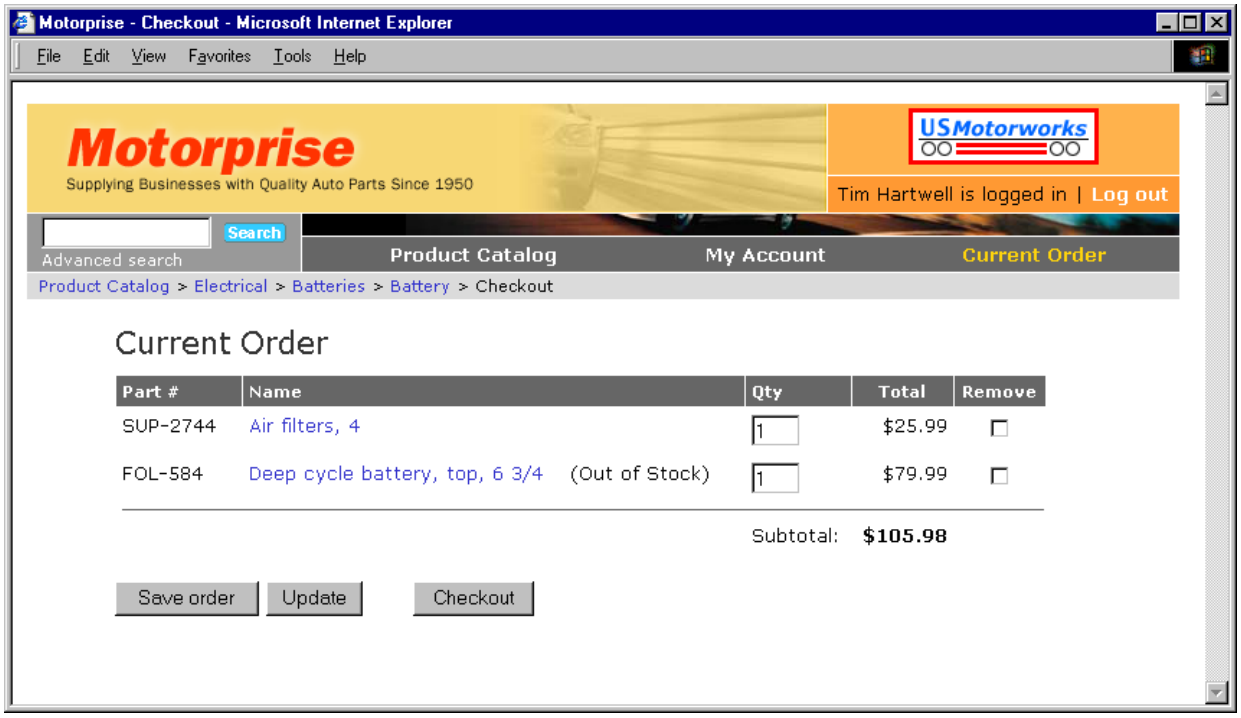
訪問した場所を収集し、それらを訪問した場所の配列に追加するために、コンポーネント /atg/commerce/catalog/CatalogNavHistoryCollector が使用されています。Motorprise では、訪問した場所は、製品カタログの製品およびカテゴリを表すリポジトリ項目です。/Motorprise/en/common/breadcrumbs.jsp からの次のスニペットが CatalogNavHistoryCollector を起動します。

```
<dsp:droplet name="/atg/dynamo/droplet/Switch">
  <dsp:param name="value" param="no_new_crumb"/>
  <dsp:oparam name="true">
  </dsp:oparam>
  <dsp:oparam name="default">
     <dsp:droplet name="CatalogNavHistoryCollector">
        <dsp:param name="navAction" param="navAction"/>
        <dsp:param name="item" param="element"/>
     </dsp:droplet>
  </dsp:oparam>
</dsp:droplet>
```

このコード・スニペットには、留意すべき点が 2 つあります。これらは両方とも Motorprise 実装固有ですが、任意のサイトに適用できます。第 1 に、必須パラメータ navCount が CatalogNavHistoryCollector に渡されていません。navCount パラメータは、このスニペットを起動するページ内で設定されます。他の JSP からサーブレットとして起動される JSP はコール元のサブスコープ内にあるため、コール元と同じパラメータにアクセスできます。

第 2 に、no_new_crumb パラメータを使用して、スニペットを起動するかどうかを決定しています。これは、現在の場所を NavHistory に追加するかどうかを決定する、ページに渡されるパラメータのスイッチにすぎません。しかし、これは、カタログ項目を表していないページのナビゲーションの対処方法を示しています。たとえば、検索ページ、ショッピング・カートおよびユーザー・プロファイル・ページは、通常のカタログ・ページのように NavHistory に追加されません。

このパラメータのもう 1 つの用途には、ユーザーがカタログを一時的に離れてもすぐに戻る、製品カタログ内の特定のインスタンスが含まれます。このような状況では、no_new_crumb パラメータを使用したブレッドクラムの特別な実装が使用されます。たとえば、ユーザーが現在のオーダーに品目を追加するとリストに「Checkout」ブレッドクラムが追加されますが、ユーザーはカタログの任意の部分にすぐに戻ることができます。ユーザーがカタログに戻ると、ブレッドクラムリストに「Checkout」値は含まれません。



ユーザーがオーダーに品目を追加するとカタログ・ブレッドクラムに「Checkout」が追加される。

ナビゲーション履歴の表示

プロパティ/atg/commerce/catalog/CatalogNavHistory.navHistory は、場所の LinkedList です。CatalogNavHistoryCollector は、前述の項に記載されているようにこのリストに移入します。breadcrumbs.jsp からの次のスニペットは、Motorprise 内でのナビゲーション履歴の表示方法を示しています。ForEach コンポーネントが NavHistory リストを反復し、リスト内の各項目に対してリンクが作成されます。JSP 内のコメントは、その動作からの変化を説明しています。

```
<%
/* ------------------------------------------------
 * use the ForEach droplet to render the navHistory array.
 * ------------------------------------------------ */
%>
<dsp:droplet name="/atg/dynamo/droplet/Switch">
  <dsp:param name="value" param="displaybreadcrumbs"/>
  <dsp:oparam name="true">

    <dsp:droplet name="/atg/dynamo/droplet/ForEach">
      <dsp:param bean="CatalogNavHistory.navHistory" name="array"/>
      <dsp:param name="elementName" value="crumb"/>
      <dsp:oparam name="output">

        <dsp:droplet name="/atg/dynamo/droplet/Switch">
          <%
```

```
  /* ------------------------------------------------
   * We want to put a separator between the items in the navHistory. In
   * this example we put | sign between them.  We use a switch droplet to
   * identify the first item in the array because we don't want to render
   * a separator, but a link to Store Home before the first item.
   * ------------------------------------------------ */
  %>
  <dsp:param name="value" param="count"/>

  <dsp:oparam name="1">
      <dsp:a href="../home.jsp">Product Catalog</dsp:a> &gt;
  </dsp:oparam>

  <dsp:oparam name="default">
    &gt;
  </dsp:oparam>

</dsp:droplet>

<dsp:droplet name="/atg/dynamo/droplet/IsNull">
  <dsp:param name="value" param="crumb"/>

  <dsp:oparam name="true">
     element is null
  </dsp:oparam>

  <dsp:oparam name="false">
    <dsp:droplet name="/atg/dynamo/droplet/Switch">
      <%
      /* ------------------------------------------------
       * Use a switch droplet to compare size to count. When
       * they are the same, then we are on the last item in
       * array iterated by the ForEach.
       * ------------------------------------------------ */
      %>
      <dsp:param name="value" param="size"/>

      <dsp:getvalueof id="countParam" idtype="Integer" param="count">
      <dsp:oparam name="<%=countParam.toString()%>">
        <dsp:droplet name="/atg/dynamo/droplet/Switch">
          <%
          /* ------------------------------------------------
           * The last item in the list is generally the item we are
           * currently visiting and should therefore not be a link.
           * In some cases, when we do not want to add a new breadcrumb,
           * we want the last item to be a link.  We do this on the
           * shopping cart page, search page, and others.  This is
           * indicated by the "no_new_crumb" parameter.
           * ------------------------------------------------ */
```

```
            %>
            <dsp:param name="value" param="no_new_crumb"/>

            <dsp:oparam name="true">
          <dsp:getvalueof id="urlStr" idtype="java.lang.String" param=
            "crumb.template.url">
              <dsp:a page="<%=urlStr%>">
                <dsp:param name="id" param="crumb.repositoryId"/>
                <dsp:param name="navAction" value="pop"/>
                <dsp:param name="Item" param="crumb"/>
                <dsp:valueof param="crumb.displayName">No name</dsp:valueof>
          </dsp:a>
          </dsp:getvalueof>
              </dsp:oparam>

              <dsp:oparam name="default">
                <dsp:valueof param="crumb.displayName"/>
              </dsp:oparam>
            </dsp:droplet>
          </dsp:oparam>
          </dsp:getvalueof>

          <dsp:oparam name="default">
         <dsp:getvalueof id="urlStr" idtype="java.lang.String" param=
           "crumb.template.url">
            <dsp:a page="<%=urlStr%>">
        <dsp:param name="id" param="crumb.repositoryId"/>
        <dsp:param name="navAction" value="pop"/>
        <dsp:param name="Item" param="crumb"/>
        <dsp:valueof param="crumb.displayName">No name</dsp:valueof>
           </dsp:a>
           </dsp:getvalueof>
          </dsp:oparam>
        </dsp:droplet>

      </dsp:oparam>
    </dsp:droplet>

  </dsp:oparam>
</dsp:droplet> <%/* end ForEach */%>
```

異なるブレッドクラムのセットの表示

ユーザーが異なるナビゲーション・コンテキストから特定のページに到着する場合があります。cart.jsp からの次のコード・スニペットでは、breadcrumbsドロップレットが、noCrumbs パラメータを調べる switchドロップレットで囲まれています。この switch により、状況のコンテキストに応じて、異なるブレッドクラムのセットを表示できます。ユーザーが製品ページから「Add to Order」を選択すると、noCrumbs パラメータが現在のオーダー・ページに渡されます。noCrumbs の値が false の場合、製品ページから該当のページに到着し、

製品カタログからのナビゲーション履歴が表示されます。それ以外の場合は、現在のオーダーが選択されたことを示す新しいブレッドクラムが表示されます。

```
<dsp:droplet name="Switch">
   <dsp:param name="value" param="noCrumbs"/>
   <dsp:oparam name="false">
      <dsp:include page="../common/breadcrumbs.jsp" flush="true"><dsp:param
         name="displaybreadcrumbs" value="true"/><dsp:param name="no_new_crumb"
         value="true"/></dsp:include> &gt; Checkout
   </dsp:oparam>
   <dsp:oparam name="default">
      <dsp:param name="noCrumbs" value="true"/>
        Current Order
   </dsp:oparam>
</dsp:droplet>
```

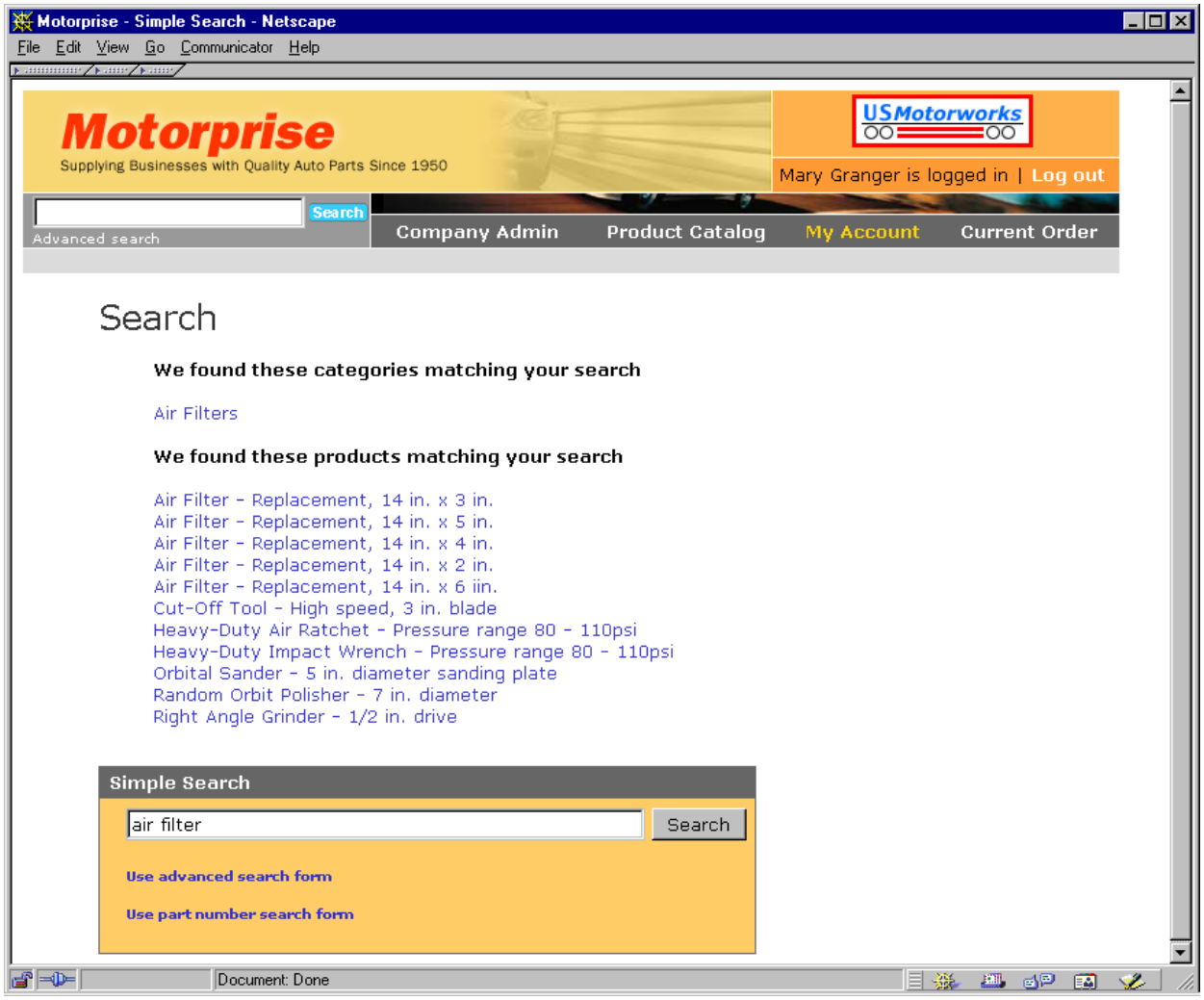
特定のページが多くの異なるコンテキストで表示される、定期オーダーの類似の実装が作成されています。

検索

検索機能により、顧客はある一連の基準を満たす製品を見つけることができます。Motorprise ストアには、単純検索および拡張検索という 2 種類の検索が実装されています。単純検索では、製品とカタログ項目の displayName プロパティおよび description プロパティに対するキーワードや単純なテキスト検索により、カタログを検索できます。つまり、検索によって製品とカテゴリの両方が返されます。拡張検索では、製造業者、カテゴリ、部品番号など、追加の製品属性を使用して検索することもできます。拡張検索では、製品のみが返されます。どちらの検索も大文字と小文字は区別されません。

単純検索

単純検索では、テキストを入力してカタログを検索します。たとえば、「air filter」を検索できます。入力は完全文字列として検索されます（たとえば、前述の問合せは、「air」または「filter」ではなく、「air filter」に基づいて行われます）。SQL リポジトリ（製品カタログ）への問合せを作成し、結果セットを返すためには、SearchFormHandler が使用されています。製品が見つからない場合は、空のセットが返されます。結果セットは、任意の形式で表示できるリポジトリ項目のコレクションです。単純検索では、製品とカテゴリの両方が返されます。



Motorprise における「air filter」の検索

ページが実行する処理を定義するためにプロパティまたは構成ファイルが使用され、実行する検索のタイプを指定するためにブール型の引数が使用されています。プロパティ名は、検索するカタログ・プロパティを指定します。単一の構成可能なフォーム・ハンドラによりカタログ検索は簡単化されますが、これですべての検索要件をサポートする必要があります。構成の変更だけでは対処できないカスタム検索がストアで必要な場合は、このフォーム・ハンドラを簡単に拡張することもできれば、別のフォーム・ハンドラを作成することもできます。

Motorprise 内の検索機能は、JSP、JSP フラグメントおよび Nucleus コンポーネント構成ファイル（プロパティ・ファイル）の組合せで実装されています。プロパティ・ファイルでは、クラス `atg.commerce.catalog.SearchFormHandler` の Nucleus コンポーネントを作成および構成して、検索を実行します。単純検索機能により、ユーザーは文字列を入力して、キーワードに一致する製品またはカテゴリを結果として取得できます。

製品には、`keywords`、`description`、`longDescription`、`displayName` など、複数のプロパティがあります。`keywords` プロパティは複数値プロパティで、その他は単一値プロパティです。そのため、どの製品も複数のキーワードが関連付けられている可能性があります。キーワード検索では、すべてのキーワード・プロパティで文字列入力が検索されます。テキスト検索では、`description` および `displayName` で文字列

入力が検索されますが、キーワード・プロパティは検索されません。デフォルトでは、各問合せでキーワードおよびテキスト検索が行われます。つまり、キーワードに加えて、摘要と表示名も問合せの対象です。

SimpleSearchFragment.jsp

検索ページ simple_search.jsp 以外の他のストア・ページからも検索できる必要があります。そのため、検索が必要な任意のページに埋め込まれる検索テキスト入力フィールドおよび「Search」ボタンを定義する、小さな検索フォーム・フラグメント/en/search/SimpleSearchFragment.jsp が作成されています。このフラグメントは Motorprise ヘッダーに埋め込まれているため、ユーザーはどのページからでも検索を行うことができます。

検索フラグメントが Motorprise ヘッダーに埋め込まれているため、ユーザーはどのページからでも検索できる

このフラグメントは、searchInput プロパティを構成し、検索ハンドラを起動するために、CatalogSearch コンポーネントをインポートします。

最後に、フォームは、FormAction パラメータを介して渡されるページ名にリダイレクトされます。このページ・フラグメントでは、検索フォームのアクションとして使用されるページ・パラメータを宣言します。つまり、FormAction パラメータは、検索フォームの submit メソッドが起動されるときにユーザーがリダイレクトされるページのファイル名です。

次の例は、SimpleSearchFragment.jsp を示しています。

```
<%/* This page fragment is intended to be embedded on any page that
     requires a search capability.  */%>


<dsp:importbean bean="/atg/commerce/catalog/CatalogSearch"/>

<dsp:form action="simple_search.jsp" method="POST">
<table cellspacing="0" cellpadding="0" border="0" bgcolor="#999999" width=100%>

  <tr><td colspan=3><dsp:img src="../images/d.gif" vspace="0"/></td></td></tr>
  <tr>
    <td><dsp:img src="../images/d.gif" hspace="3"/></td>
```

```
    <td width=100>
    <!-- form elements -->
    <input name="repositoryKey" type="hidden" value="<dsp:valueof
      bean='/OriginatingRequest.requestLocale.locale'/>">
    <dsp:input bean="CatalogSearch.searchInput" size="15" type="text" value=""/>
    </td>
     <!--  use this hidden form tag to make sure the search handler is invoked if
       someone does not hit the submit button -->
     <td align=center>
    <dsp:input bean="CatalogSearch.search" type="hidden" value="Search"/>
    <dsp:input type="image" src="../images/motorprise-search.gif"
      bean="CatalogSearch.search" name="search" value="Search" border="0"/> </td>
    <!-- end form elements -->

  </tr>
  <tr>
    <td><dsp:img src="../images/d.gif" hspace="3"/></td>
    <td colspan=2><dsp:a href="advanced_search.jsp"><font color="#FFFFFF"><span
      class=smallw>Advanced search</span></font></dsp:a></td>
  </tr>
  <tr><td colspan=3><dsp:img src="../images/d.gif" vspace="0"/></td></tr>


</dsp:form>
</table>
```

hidden 入力タグは、単一の入力フィールドを持つフォームの HTML 対策です。hidden 入力により、テキスト入力フィールドで[Enter]キーを押すことによって `submit` メソッドを起動できます。

この hidden タグがない場合は、ユーザーは「**Search**」ボタンをクリックする必要があります。

simple_search.jsp

`simple_search.jsp` は、独自の機能を提供する様々な JSP コード・フラグメントで構築されるページです。`simple_search.jsp` は `SearchResults.jsp` を使用して、検索の結果を表示します。

最初に、`SearchResults.jsp` を含めることによって、ページの上部に検索結果を表示します。実際の検索を実行するには、`CatalogSearch` コンポーネントを使用します。ユーザーが検索を発行すると、ハンドラ検索メソッドが起動されます。これが基本的に `catalogSearch` の `ItemType` プロパティにより検索結果を書き込みます。

次の例は、`simple_search.jsp` を示しています。

```
<%@ taglib uri="dsp" prefix="dsp" %>
<dsp:page>

<dsp:importbean bean="/atg/dynamo/droplet/ForEach"/>
<dsp:importbean bean="/atg/commerce/catalog/CatalogSearch"/>
<dsp:importbean bean="/atg/dynamo/droplet/Switch"/>

<DECLAREPARAM NAME="noCrumbs"
```

```
   CLASS="java.lang.String"
   DESCRIPTION="This is for deciding what kind of breadcrumbs to display.
   If this is true, then breadcrumbs will show: Store Home|Search,
   instead of nav history. Default is false."
   OPTIONAL>


<dsp:include page="../common/HeadBody.jsp" flush="true"><dsp:param
  name="pagetitle" value=" Simple Search"/></dsp:include>
<table border=0 cellpadding=0 cellspacing=0 width=800>
<tr>
  <td colspan=2><dsp:include page="../common/BrandNav.jsp"
    flush="true"></dsp:include></td>
</tr>
<tr bgcolor="#DBDBDB" >
  <td colspan=2 height=18> 
  <!-- breadcrumbs -->
  <span class="small">
  <dsp:droplet name="Switch">
    <dsp:param name="value" param="noCrumbs"/>
      <dsp:oparam name="true"><dsp:a href="../home.jsp">Product Catalog</dsp:a>
          Simple Search</dsp:oparam>
      <dsp:oparam name="default"><dsp:param name="noCrumbs" value="false"/>
        <dsp:include page="../common/breadcrumbs.jsp" flush="true"><dsp:param
          name="displaybreadcrumbs" value="true"/><dsp:param name="no_new_crumb"
            value="true"/></dsp:include>
      </dsp:oparam>
  </dsp:droplet></span>
  </td>
</tr>


<tr>
  <td width=55><dsp:img src="../images/d.gif" hspace="27"/></td>
  <td valign="top" width=745><br><span class="big">Search</span></td>
</tr>

<tr>
  <td width="40"><dsp:img src="../images/d.gif"/></td>
  <td><br>
    <dsp:getvalueof id="pval0" bean="CatalogSearch.searchResultsByItemType">
      <dsp:include page="SearchResults.jsp" flush="true"><dsp:param
      name="ResultArray" value="<%=pval0%>"/></dsp:include></dsp:getvalueof>
  </td>
</tr>

<tr><td colspan=2><dsp:img src="../images/d.gif" vspace="6"/></td></tr>

<tr>
```

```
<td width="40"><dsp:img src="../images/d.gif"/></td>
<td><!-- simple search box -->
  <table bgcolor="#FFCC66" border=0 cellpadding=0 cellspacing=0>
    <tr>
      <td colspan=3>
        <table width=100% cellpadding=4 cellspacing=0 border=0>
        <tr><td class=box-top> Simple Search</td></tr>
        </table>
      </td>
    </tr>

    <tr><td bgcolor="#666666"><dsp:img src="../images/d.gif" width="1"/></td>
        <td>
          <dsp:form action="simple_search.jsp" method="POST">
          <table width=100% cellpadding=6 cellspacing=0 border=0>
          <tr>
            <td></td>
            <td bgcolor="#ffcc66">
            <input name="repositoryKey" type="hidden" value="<dsp:valueof
              bean="/OriginatingRequest.requestLocale.locale"/>">
            <dsp:input bean="CatalogSearch.searchInput" size="30" type="text"/>
            <input name="noCrumbs" type="hidden" value="<dsp:valueof
              param="noCrumbs"/>">
            <!--  use this hidden form tag to make sure the search handler is
              invoked if someone does not hit the submit button -->
            <dsp:input bean="CatalogSearch.search" type="hidden"
              value="Search"/>
            <dsp:input bean="CatalogSearch.search" type="submit"
              value="Search"/><br><!--<span class="help">Separate words or
                phrases by <b>AND</b> or <b>OR</b></span>-->
            <p>
            <span class=smallb><dsp:a href="advanced_search.jsp">
              <dsp:param name="noCrumbs" param="noCrumbs"/>
              Use advanced search form</span></dsp:a>
            </td>
          </tr>
          <tr>
            <td></td>
            <td>
            <span class=smallb><dsp:a href="part_number_search.jsp">
             <dsp:param name="noCrumbs" param="noCrumbs"/>
               Use part number search form</dsp:a></span>
            </td>
          </tr>
          </table>
          </dsp:form>
        </td>
        <td bgcolor="#666666"><dsp:img src="../images/d.gif" width="1"/></td>
    </tr>
```

```
      <tr><td bgcolor="#666666" colspan=3><dsp:img src=
        "../images/d.gif"/></td></tr>
      </table>
    </td>
  </tr>
  </table>


</body>
</html>
</dsp:page>
```

SearchResults.jsp

simple_search.jsp からの検索結果を反復および表示するために、SearchResults ページ・フラグメントが使用されています。ForEach コンポーネントには、様々な種類の複数値オブジェクトを格納できるパラメータ配列が送られます。

この場合のタイプは、SearchFormHandler コンポーネントの成果である java.util.Map です。

マップは、コンポーネントの構成ファイルの itemTypes 値で指定される値ごとに 1 つのエントリを持ちます。SimpleSearch の場合、itemTypes は category および product です。

マップは、検索結果配列およびそれに関連する itemType 識別子のコレクションです。マップ内でカテゴリおよび製品に関連付けられた値は、カテゴリまたは製品品目の配列です。または、そのキーに対して結果が見つからなかった場合は null です。

カテゴリおよび製品のマップ要素は、最も外側の ForEach ドロップレット・タグ内の標準の ForEach コンポーネントを使用して、マップを反復することによって抽出されます。マップの各要素に対して Switch コンポーネントが起動されて、カテゴリおよび製品要素を識別し、これらがページ上にレンダリングされる順序に影響します。

カテゴリおよび製品の検索結果配列が取得されたら、その配列の各要素を反復して結果を表示する必要があります。空の結果配列を検出するためには、別の Switch コンポーネントが使用されています。これは dsp:oparam を unset にする Switch 機能で、これにより、空の入力配列が識別され、指定のタイプの項目が見つからなかったことを示すメッセージがレンダリングされます。配列が null でない場合は、Switch のデフォルトの dsp:oparam タグがレンダリングされます。ここでは、別の ForEach コンポーネント・コールが配列を反復し、その各項目を表示します。

```
<%@ taglib uri="dsp" prefix="dsp" %>
<dsp:page>


<%
/* ----------------------------------------------------------------
This JSP droplet displays the contents of search
that potentially returns both category and product repository items.
The one paramater, ResultArray, accepts a HashMap that contains
elements with the keys "category" and "product".  The values of these
keys are collections of category or product repository items found in
the search.
----------------------------------------------------------------*/
```

```
%>


<DECLAREPARAM NAME="ResultArray"
              CLASS="java.util.Map"
              DESCRIPTION="Array of Search Results">


<dsp:importbean bean="/atg/dynamo/droplet/Switch"/>
<dsp:importbean bean="/atg/dynamo/droplet/IsEmpty"/>
<dsp:importbean bean="/atg/dynamo/droplet/ForEach"/>
<dsp:importbean bean="/atg/dynamo/droplet/RQLQueryForEach"/>


<dsp:droplet name="ForEach">
  <dsp:param name="array" param="ResultArray"/>

  <%/*Each item in this array is a Collection of Categories or Products...*/%>
  <dsp:param name="elementName" value="ResultCollection"/>

  <dsp:oparam name="output">
    <dsp:droplet name="Switch">

      <%--The key tells us if this is a Collection of Products or Categories: --%>
      <dsp:param name="value" param="key"/>

      <%--For the list of CATEGORIES: --%>
      <dsp:oparam name="category">

        <blockquote>

        <dsp:droplet name="Switch">
          <dsp:param name="value" param="ResultCollection"/>
          <dsp:oparam name="default">
            <p>

            <%-- For each Category in the Collection: --%>
            <dsp:droplet name="ForEach">
              <dsp:param name="array" param="ResultCollection"/>
              <dsp:param name="sortProperties" value="+displayName"/>
              <dsp:param name="elementName" value="Category"/>
              <dsp:oparam name="outputStart">
                <b>We found these categories matching your search</b>
                <p>
              </dsp:oparam>
              <dsp:oparam name="output">

                <%-- Display a link to the Category: --%>
          <dsp:getvalueof id="urlStr" idtype="java.lang.String"
```

```
        param="Category.template.url">
            <dsp:a page="<%=urlStr%>">
              <dsp:param name="id" param="Category.repositoryId"/>
              <dsp:param name="navAction" value="jump"/>
              <dsp:param name="Item" param="Category"/>
              <dsp:valueof param="Category.displayName">No name</dsp:valueof>
    </dsp:a>
    </dsp:getvalueof>
          <br>
        </dsp:oparam>
        <dsp:oparam name="empty">
          <b>There are no categories matching your search</b>
          <p>
        </dsp:oparam>
      </dsp:droplet>
    </dsp:oparam>
    <%-- If NO Categories returned by the search: --%>
    <dsp:oparam name="unset">
      No category items in the catalog could be found that match your query
    </dsp:oparam>
  </dsp:droplet><%/*ForEach Category*/%>

  </blockquote>
  <P>
</dsp:oparam>

<%-- For the list of PRODUCTS: --%>
<dsp:oparam name="product">
  <blockquote><p>

  <dsp:droplet name="Switch">
    <dsp:param name="value" param="ResultCollection"/>

    <dsp:oparam name="default">

      <%/*For each Product in the Collection: */%>
      <dsp:droplet name="ForEach">
        <dsp:param name="array" param="ResultCollection"/>
        <dsp:param name="sortProperties" value="+displayName"/>
        <dsp:param name="elementName" value="Product"/>
        <dsp:oparam name="outputStart">
          <p>
          <b>We found these products matching your search</b>
          <p>
        </dsp:oparam>
        <dsp:oparam name="output">
          <%-- Display a link to the Product: --%>
    <dsp:getvalueof id="urlStr" idtype="java.lang.String" param=
      "Product.template.url">
```

```
            <dsp:a page="<%=urlStr%>">
              <dsp:param name="id" param="Product.repositoryId"/>
              <dsp:param name="navAction" value="jump"/>
              <dsp:param name="Item" param="Product"/>
              <dsp:valueof param="Product.displayName">No name
              </dsp:valueof> - <dsp:valueof
                param="Product.description"/>
          </dsp:a>
          </dsp:getvalueof>
                <br>
              </dsp:oparam>
              <dsp:oparam name="empty">
                <b>There are no products matching your search</b>
                <p>
              </dsp:oparam>

            </dsp:droplet> <%/*ForEach Product*/%>

          </dsp:oparam>

          <%/*If NO Products returned by the search: */%>
          <dsp:oparam name="unset">
            No product items in the catalog could be found that match your
              query<p>
          </dsp:oparam>

        </dsp:droplet>
        </blockquote><P>
      </dsp:oparam>
    </dsp:droplet>

  </dsp:oparam>

</dsp:droplet> <%/*ForEach Item returned by Search */%>
</dsp:page>
```

拡張検索

単純検索機能のキーワードおよびテキスト検索に加えて、拡張検索機能では、製品のプロパティとして定義されている部品番号や製造業者で製品を検索できます。拡張検索は、部品番号検索と、製造業者およびカテゴリを使用した検索の 2 種類に分けられます。

部品番号検索

部品番号は Product 項目のプロパティではなく sku 項目のプロパティであるため、部品番号を検索するには atg/projects/b2bstore/catalog/PartNumberSearchFormHandler という別のコンポーネントを使用します。このコンポーネントはカスタム・フォーム・ハンドラ CatalogSearchFormHandler を使用し、AdvProductSearch とは異なるプロパティを使用します。

catalogSearchFormHandler のプロパティ・ファイルを、次に示します。

```
# Allow users to search by part number
# Part number exists on a sku
$class=atg.commerce.catalog.custom.CatalogSearchFormHandler
$scope=request
catalogTools=/atg/commerce/catalog/CatalogTools
doTextSearch=true
textSearchPropertyNames=manufacturer_part_number
itemTypes=sku
repositories=/atg/commerce/catalog/ProductCatalog
```

次のコードは、part_number_search.jsp 内で部品番号検索を行うためのコードです。

```
<dsp:form action="part_number_search.jsp" method="POST">
<table width="100%" border="0" cellspacing="0" cellpadding="4" bgcolor="#FFCC66">
   <tr><td colspan=2><dsp:img src= "../images/d.gif" height="10"/></td></tr>
   <tr valign="top">
      <td width="15%" align="right">
      <span class="smallb">Part#</span>
      </td>
      <td>
         <dsp:input bean="PartNumberSearchFormHandler.searchInput" size="25"
            type="text"/> 
            <br> <span class="small">Use <b>*</b> for partial part number
               search.</span>
      </td>
   </tr>
   <tr>
      <td> </td>
      <td>
      <input name="repositoryKey" type="hidden" value="<dsp:valueof
         bean="/OriginatingRequest.requestLocale.locale"/>">
      <input name="noCrumbs" type="hidden" value="<dsp:valueof
         param="noCrumbs"/>"><br>
      <dsp:input bean="PartNumberSearchFormHandler.search" type="hidden"
        value="Search"/>
      <dsp:input bean="PartNumberSearchFormHandler.search" type="submit"
        value="Search"/>
</dsp:form></td>
```

PartNumberSearchFormHandler コンポーネントの SearchInput プロパティは、ユーザーから入力された部品番号を使用して設定されます。これに基づいて、SKU の manufacturer_part_number プロパティが検索されます。

part_number_search.jsp は、見つかった製品の SKU の配列が含まれる検索結果を渡して、PartNumberSearchResults.jsp フラグメントを起動します。次のコードは、これが含まれる part_number_search.jsp 内のコード・フラグメントです。

```
<dsp:getvalueof id="pval0" idtype="java.util.List"
   bean="PartNumberSearchFormHandler.searchResults">

   <dsp:include page="PartNumberSearchResults.jsp" flush="true">
   <dsp:param name="Skus" value="<%=pval0%>"/>
   </dsp:include>

   </dsp:getvalueof>
```

指定の部品番号の SKU を取得したら、PartNumberSearchResults.jsp でその SKU に対応する製品をフェッチします。RQLQueryForEach を使用して各 SKU を反復し、その SKU に対応するすべての製品を取得して、それらを表示します。

見つかった製品を表示する PartNumberSearchResults.jsp からのスニペットを、次に示します。

```
<dsp:droplet name="ForEach">
  <dsp:param name="array" param="Skus"/>
  <dsp:param name="elementName" value="element"/>
  <dsp:oparam name="outputStart">
   <b>We found these products matching your search</b>
   <br><br>
  </dsp:oparam>
  <dsp:oparam name="output">

  <dsp:droplet name="RQLQueryForEach">
    <dsp:param name="skuId" param="element"/>
    <dsp:param name="repository" bean="/atg/commerce/catalog/ProductCatalog" />
    <dsp:param name="itemDescriptor" value="product"/>
    <dsp:param name="queryRQL" value="childSKUs INCLUDES :skuId  "/>
    <dsp:param bean="/atg/dynamo/transaction/TransactionManager"
      name="transactionManager"/>
    <dsp:oparam name="output">
      <%-- Display a link to the element: --%>
      <dsp:getvalueof id="urlStr" idtype="java.lang.String"
        param="element.template.url">
      <dsp:a page="<%=urlStr%>">
        <dsp:param name="id" param="element.repositoryId"/>
     <dsp:param name="navAction" value="jump"/>
     <dsp:param name="Item" param="element"/>
     <dsp:valueof param="element.displayName">No name</dsp:valueof> -
        <dsp:valueof param="element.description"/>
      </dsp:a>
      </dsp:getvalueof>
      <br>
     </dsp:oparam>
  </dsp:droplet>
  </dsp:oparam>
  <dsp:oparam name="empty">
    <b>There are no products matching your search</b>
```

```
    <p>
  </dsp:oparam>
</dsp:droplet>
```

製造業者およびテキスト入力検索

この検索では AdvProductSearch コンポーネントが使用されます。

拡張検索機能では SearchFormHandler を使用しますが、単純検索の Nucleus コンポーネントより多少複雑なプロパティ・ファイルが必要です。拡張検索の追加の検索基準は製品にのみ基づいているため、拡張検索は製品のみを返します。

/atg/commerce/catalog/AdvProductSearch.properties

```
$class=atg.commerce.catalog.SearchFormHandler
$scope=session
doKeywordSearch=true
keywordsPropertyNames=keywords
doTextSearch=true
textSearchPropertyNames=description,displayName
doAdvancedSearch=true
advancedSearchPropertyNames=childSKUs
doHierarchicalSearch=true
ancestorCategoriesPropertyName=ancestorCategories
minScore=1
catalogTools=CatalogTools
itemTypes^=CatalogTools.productItemTypes
maxResultsPerPage=20
enableCountQuery=true
```

Motorprise 構成レイヤーで、このコンポーネントに manufacturer プロパティが追加されています。

```
# /atg/commerce/catalog/AdvProductSearch
advancedSearchPropertyNames+=manufacturer
```

単純検索と同様、SearchFormHandler は session-scoped です。キーワードおよびテキスト検索は、単純検索と拡張検索の両方で同じように構成されており、どちらも同じカタログを使用します。検索（manufacturer）で列挙型が使用される場合、使用できる値がフォーム上の<select>入力タグに挿入されます。これらの値はフォームにコード化されるのではなく、SearchFormHandler の propertyValuesByType プロパティを介してカタログから取得されます。

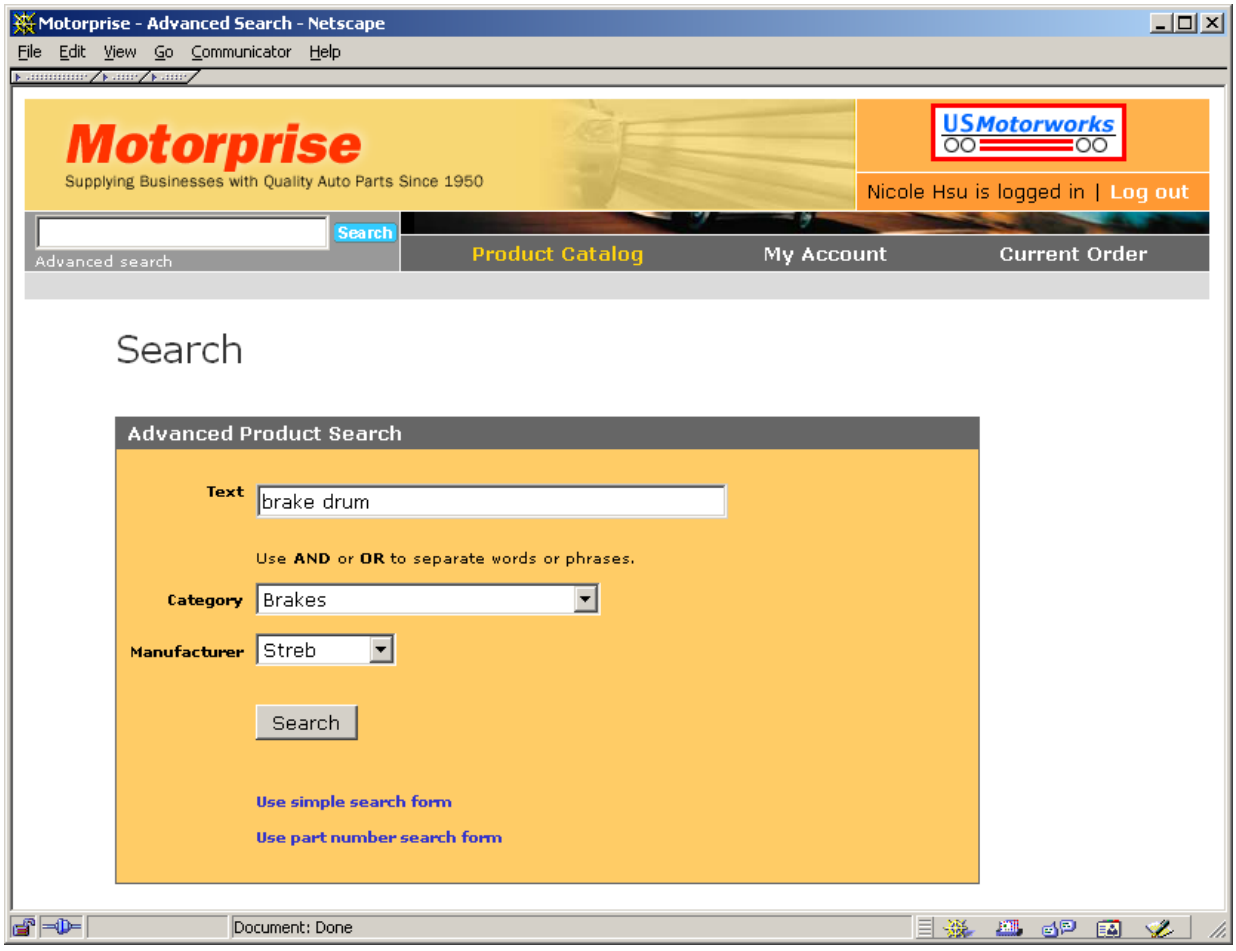
propertyValuesByType プロパティは、キーのプロパティ名、および特定のプロパティで使用可能なすべての値の配列を含むマップです。デフォルトでは、SearchFormHandler の propertyValuesByType プロパティから列挙型プロパティの情報は得られません。検索コンポーネントの advancedSearchPropertyNames プロパティに、必要な列挙型プロパティの名前を明示的に設定する必要があります。この場合の値は manufacturer です。

Motorprise で検索が実行されると、propertyValuesByType マップは 2 つのエントリを返します。1 つのエントリは category 項目タイプに対応し、もう 1 つのエントリは product 項目タイプに対応しています。これは、これらが ProductCatalog リポジトリ内で検索される 2 つの項目記述子タイプのためです。カテゴリに関連するエントリは、検索で見つかったカテゴリ・リポジトリ項目のリストとなり、同様に、製品キーには製品リポジトリ項目のリストが含まれます。

タイプ別のリポジトリ項目のリストを持つことで、リポジトリ項目の表示方法を制御できます。たとえば、適合する製品リポジトリ項目を表示する場合は product.jsp にリンクし、カテゴリの場合は category.jsp にリンクします。前述の simple_search.jsp の例で、propertyValuesByType 配列の key プロパティについて switch 文があるのはこのためです。

拡張検索（manufacturer）の検索フォームのカテゴリには、各項目の有効な選択肢が移入されます。これらの選択肢（<select>コンポーネント）は、ページにコード化されているのではなく、リポジトリからロードされるため、リポジトリが変更されてもページを変更する必要はありません。

すべてのカテゴリのリストを表示するためには、RepositoryValues サーブレット Bean が使用されています。これは入力パラメータ itemDescriptorName を受け取ります。この例では、リストするリポジトリ項目のタイプであるカテゴリに設定されています。サーブレット Bean が出力するリポジトリ項目の配列は、ForEach コンポーネントを使用してレンダリングできます。



拡張検索の使用

advanced_search.jsp 内のカテゴリ検索のコードの一部を、次に示します。

```
<!--search box starts here-->
  <table width=100% bgcolor="#FFCC66" border=0 cellpadding=0 cellspacing=0>
```

```
<tr bgcolor="#666666">
  <td bgcolor="#666666" width=1><dsp:img src="../images/d.gif"
    width="1"/></td>
  <td colspan=2>
  <table cellpadding=3 cellspacing=0 border=0>
    <tr><td class=box-top> Advanced Product Search</td></tr>
  </table>
  </td>
</tr>

<tr>
  <td bgcolor="#666666" width=1><dsp:img src="../images/d.gif"
    width="1"/></td>
  <td>
  <table width=100% bgcolor="#FFCC66" border=0 cellpadding=0 cellspacing=0>
    <tr valign=top>
      <td width=50%>
        <dsp:form action="advanced_search.jsp" method="POST">
        <table width="100%" border="0" cellspacing="0" cellpadding="4"
          bgcolor="#FFCC66">
          <tr><td colspan=2><dsp:img src= "../images/d.gif"
            height="10"/></td></tr>

          <tr valign=top>
            <td width="15%" align="right"><span class=
              "smallb">Text</span></td>
            <td><dsp:input bean="AdvProductSearch.searchInput" size="25"
              type="text"/>  <br> <span class="small">Use <b>AND</b> or
              <b>OR</b> to separate words or phrases.</span>
            </td>
          </tr>
          <tr>
            <td align="right"><span class=smallb>Category</span></td>
            <td>
            <dsp:select bean="AdvProductSearch.hierarchicalCategoryId">
              <dsp:option value=""/>-- All categories --
              <dsp:droplet name="RepositoryValues">
                <dsp:param name="itemDescriptorName" value="category"/>
                <dsp:oparam name="output">
                  <dsp:droplet name="ForEach">
                    <dsp:param name="array" param="values"/>
                    <dsp:param name="sortProperties" value="+displayName"/>
                    <dsp:oparam name="output">
                      <dsp:getvalueof id="elem"
                       idtype="atg.repository.RepositoryItem" param="element">
                        <dsp:option value="<%=elem.getRepositoryId()%>"/>
                        <dsp:valueof param="element.displayName"/>
                      </dsp:getvalueof>
                    </dsp:oparam>
```

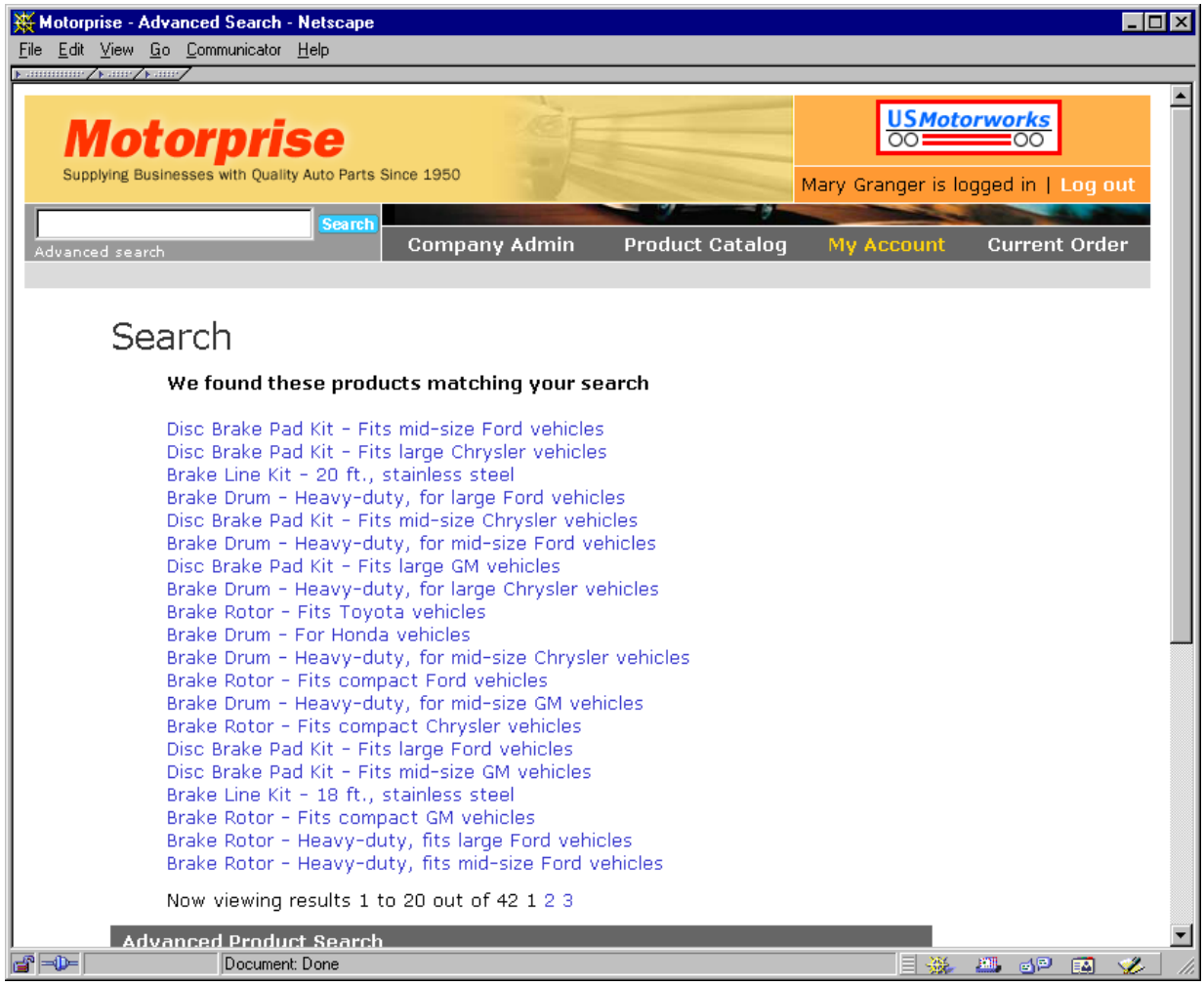
```
          </dsp:droplet>
        </dsp:oparam>
      </dsp:droplet>
    </dsp:select>
    </td>
  </tr>
  <tr>
    <td align="right"> <span class=
      smallb>Manufacturer</span></td>
    <td>
    <dsp:select bean="AdvProductSearch.propertyValues.manufacturer">
      <dsp:option value=""/>-- Any --
      <dsp:droplet name="RepositoryValues">
        <dsp:param name="itemDescriptorName" value="product"/>
        <dsp:param name="propertyName" value="manufacturer"/>
        <dsp:oparam name="output">
          <dsp:droplet name="ForEach">
            <dsp:param name="array" param="values"/>
            <dsp:param name="sortProperties" value="+displayName"/>
            <dsp:oparam name="output">

              <dsp:getvalueof id="elemName" idtype="java.lang.String"
                param="element.displayName">
                <dsp:option value="<%=elemName%>"/>
                <dsp:valueof param="element.displayName"/>
              </dsp:getvalueof>

            </dsp:oparam>
          </dsp:droplet>
        </dsp:oparam>
      </dsp:droplet>
    </dsp:select>
    </td>
  </tr>
  <tr>
    <td> </td>
    <td><br>
    <dsp:input bean="AdvProductSearch.currentResultPageNum"
      type="hidden" value="1"/>
    <input name="repositoryKey" type="hidden" value="<dsp:valueof
      bean="/OriginatingRequest.requestLocale.locale"/>">
    <input name="noCrumbs" type="hidden" value="<dsp:valueof
param="noCrumbs"/>">
    <dsp:input bean="AdvProductSearch.search" type="hidden"
      value="Search"/>
    <dsp:input bean="AdvProductSearch.search" type="submit"
      value="Search"/>
    </td>
  </tr>
```

```
<tr>
  <td colspan=2> </td>
</tr>
<tr>
  <td> </td>
  <td><span class=smallb><dsp:a href="simple_search.jsp">
        <dsp:param name="noCrumbs" param="noCrumbs"/>
        Use simple search form</dsp:a></span>
  </td>
</tr>
```

検索結果の表示

Oracle Commerce Core Commerce に付属の SearchFormHandler では、1 つの項目タイプで検索した場合のみ検索結果をナビゲートできます。拡張検索では、製品のみが検索されます。単純検索では製品とカテゴリの両方が検索されるため、ナビゲーション機能は使用されません。

SearchFormHandler を構成して拡張検索の場合は一度に 20 件の検索結果のみを返すようにし、表示されている結果セットと合計数をユーザーに伝えるテキストを含めました（単純検索では 1 ページにすべての結果がリストされます）。



拡張検索では一度に 20 件の結果が表示される。

ページ・ナビゲーションを可能にし、返される結果数を設定するためには、プロパティ `enableCountQuery` および `maxResultsPerPage` が使用されています。拡張検索および製造業者部品番号を使用する検索が、1 ページ当たり 20 件の結果のみを返すように構成します。`advanced_search.jsp` 内で次のコードを使用してページ・ナビゲーション URL を表示します。

```
<dsp:droplet name="Compare">
  <dsp:param bean="AdvProductSearch.resultSetSize" name="obj1"/>
  <dsp:param bean="AdvProductSearch.maxResultsPerPage" name="obj2"/>
  <dsp:oparam name="greaterthan">
    Now viewing results
    <b><dsp:valueof bean="AdvProductSearch.startCount"/> -
    <dsp:valueof bean="AdvProductSearch.endIndex"/></b>
    out of
    <b><dsp:valueof bean="AdvProductSearch.resultSetSize"/></b>
  </dsp:oparam>
```

```
</dsp:droplet>

<dsp:droplet name="Switch">
 <dsp:param bean="AdvProductSearch.resultPageCount" name="value"/>
 <dsp:oparam name="1">
 </dsp:oparam>

 <dsp:oparam name="default">
  <br>Results pages:
  <dsp:droplet name="Switch">
     <dsp:param name="value" bean="AdvProductSearch.currentResultPageNum"/>
     <dsp:oparam name="1">

     </dsp:oparam>
     <dsp:oparam name="default">
       <dsp:droplet name="For">
         <dsp:param name="howMany" bean="AdvProductSearch.resultPageCount"/>
         <dsp:oparam name="output">
           <dsp:droplet name="Switch">
             <dsp:param name="value" bean=
               "AdvProductSearch.currentResultPageNum"/>
             <dsp:getvalueof id="countParam" idtype="Integer" param="count">
               <dsp:oparam name="<%=countParam.toString()%>">
                 <dsp:a href="advanced_search.jsp" bean=
                   "AdvProductSearch.currentResultPageNum"
                   paramvalue="index">&lt;&lt; Previous</dsp:a>  
               </dsp:oparam>
             </dsp:getvalueof>
           </dsp:droplet>
         </dsp:oparam>
       </dsp:droplet>
     </dsp:oparam>
   </dsp:droplet>
```

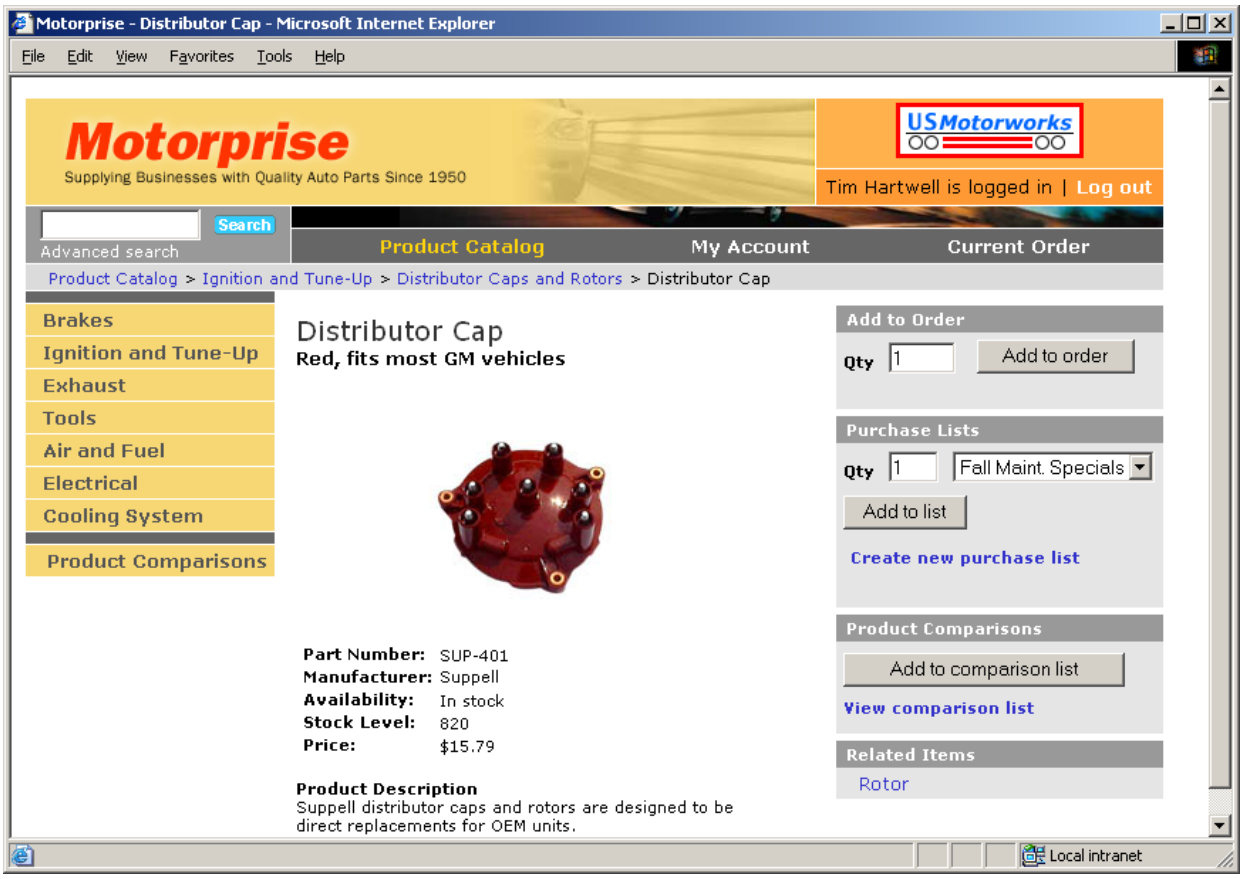
SearchFormHandler の currentResultPageNum プロパティによって、表示する結果セットが決まります。resultPageCount を反復して各ページの URL を表示します。ユーザーが任意の URL をクリックすると対応する番号が SearchFormHandler.currentResultPageNum に設定され、該当のセットが返されます。

比較

Motorprise サイトには製品比較機能が含まれており、ユーザーはこの機能を使用して様々な製品の属性を同時に確認できます。比較リストには製品ページから製品を追加できます。「Product Comparisons」ページで製品を検索し、検索結果から比較リストに追加することもできます。

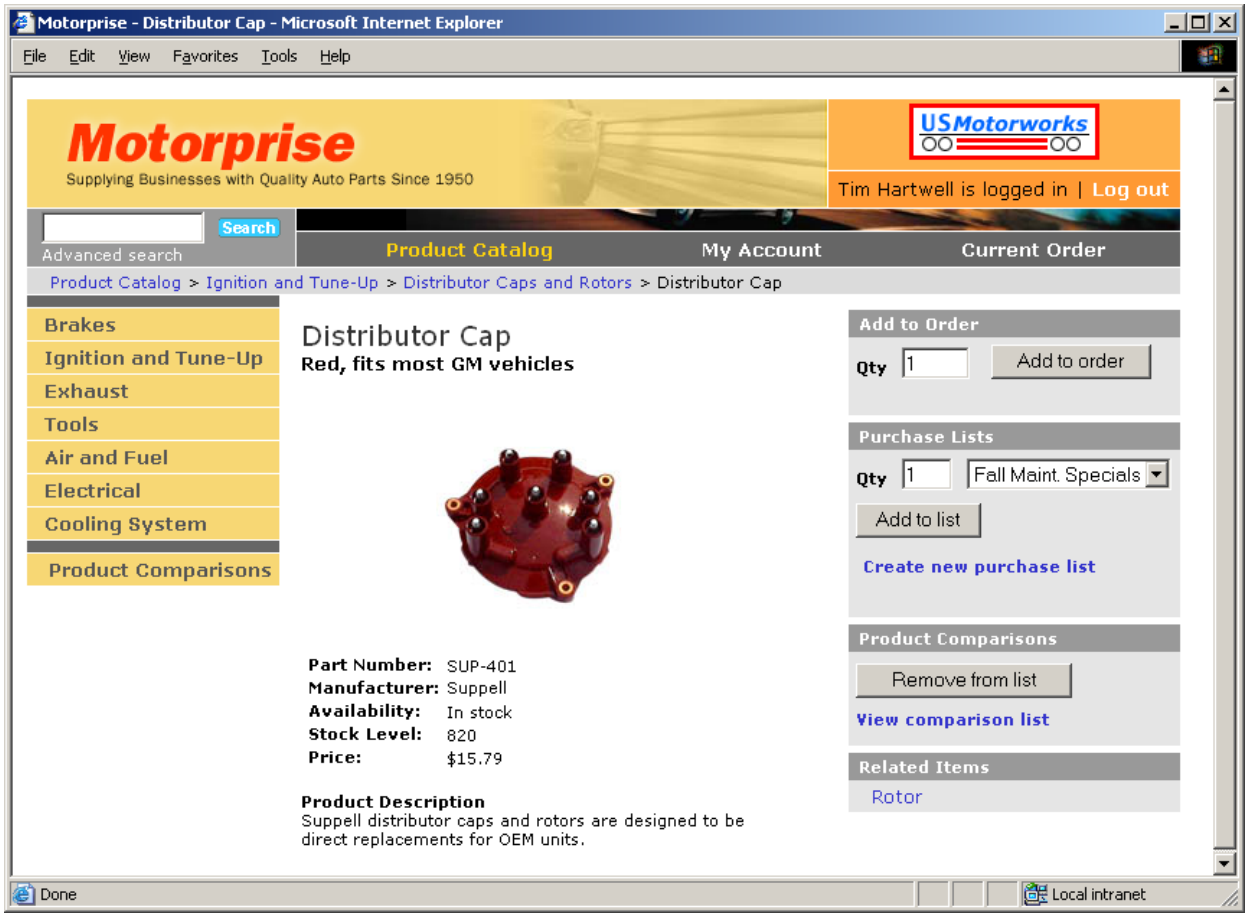
製品ページから比較リストへの製品の追加

製品ページでは、単純なフォームを使用して「**Add to comparison list**」ボタンを表示しています。ユーザーに表示されている製品を、比較する製品のリストに追加するためには、`ProductListHandler.addProduct` メソッドが使用されています。



製品ページ上の、品目の追加および比較リストの表示を行うセクション

比較リストに製品がある場合は、ページに「**View comparison list**」リンクが表示されます。ユーザーがリストに製品を追加すると、ボタンが変更されて「**Remove from list**」が表示され、`ProductListHandler.removeProduct` メソッドを使用してリストから製品を削除できるようになります。



比較リストに製品がある場合は、ボタンが変更され「Remove from list」が表示される。

製品ページ上に「Product Comparisons」セクションを実装するために使用したコードを、次に示します。

```
<dsp:form action="product.jsp" method="post">
  <tr>
    <td bgcolor="#E5E5E5">
    <input name="id" type="hidden" value="<dsp:valueof
      param="element.repositoryId"/>">
    <dsp:input bean="CartModifierFormHandler.SessionExpirationURL" type="hidden"
      value="../common/session_expired.jsp"/>
    <dsp:input bean="CartModifierFormHandler.productId"
      paramvalue="product.repositoryId" type="hidden"/>

    <dsp:droplet name="ProductListContains">
      <dsp:param bean="ProductList" name="productList"/>
      <dsp:param name="productID" param="element.repositoryId"/>
      <dsp:oparam name="true">
        <dsp:input bean="ProductListHandler.productID" paramvalue="productID"
          type="hidden"/>
```

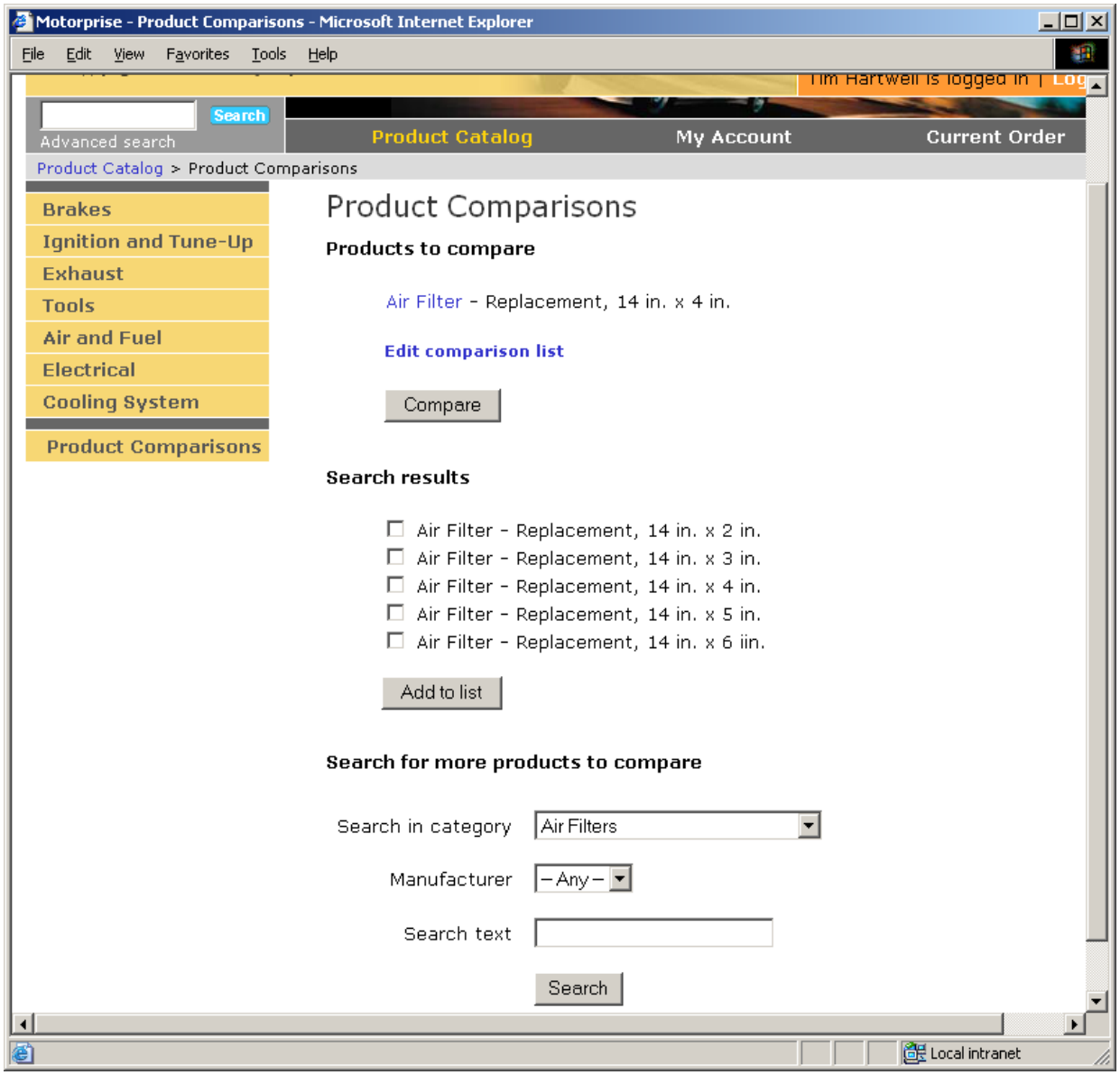
```
        <dsp:input bean="ProductListHandler.removeProduct" type="submit"
          value="Remove from list"/>
       </dsp:oparam>
       <dsp:oparam name="false">
         <dsp:input bean="ProductListHandler.productID" paramvalue="productID"
           type="hidden"/>
         <dsp:input bean="ProductListHandler.addProduct" type="submit" value="Add
           to comparison list"/>
        </dsp:oparam>
      </dsp:droplet>
  </td>
</tr>
<!--check to see if there are items in comparison list-->
<dsp:droplet name="IsEmpty">
  <dsp:param bean="ProductList.items" name="value"/>
  <dsp:oparam name="false">
<tr>
  <td><span class=smallb><dsp:a href="compare.jsp">View comparison
    list</dsp:a></td>
</tr>
  </dsp:oparam>
  </dsp:droplet>
  </dsp:form>
```

「Product Comparisons」ページの使用

左側のカタログ・ナビゲーション・バー上のカタログのリストの下には**「Product Comparisons」**リンクがあり、これをクリックすると「Product Comparisons」ページに移動して、カテゴリ、製造業者またはテキストで比較する製品を検索できます。

「Product Comparisons」ページは、3 つのセクションで構成されています。

- 比較する製品のリスト（ユーザーが製品を追加した場合）
- 検索フォーム
- 検索結果（検索が実行された場合）



「Product Comparisons」ページ

比較する製品の検索

検索フォームを使用してカテゴリ、製造業者、テキストまたはこれらの組合せを入力することによって、製品を検索できます。検索結果は検索ボックスの上にチェック・ボックスとともに表示され、これらを比較リストに追加できます。

検索フォームには `atg.commerce.catalog.custom.SearchFormHandler` のインスタンスが使用され、それが比較用に構成されています。製品の検索、名前および説明の検索、ならびに特定のカテゴリ内の検索を可能にする階層検索を行えるように設定されています。

製品ページと同じ方法を使用して、結果リストから比較リストに製品が追加されます。session-scoped の比較リストとは異なり、検索結果リストは一時的な手順のため request-scoped です。選択された製品が比較リストに追加されると、検索結果は消去されます。選択された品目の削除、あるいは比較リスト全体の消去をユーザーが実行できるようにするため、ProductListHandler が使用されています。

ProductListHandler の使用の詳細は、『ATG Web Commerce ストア設定コマース・ガイド』の「*製品比較機能の実装*」を参照してください。

比較結果の表示

ユーザーが「**Compare**」ボタンをクリックすると、結果が部品番号、製品名、摘要、製造業者、価格および可用性のリストとして表示されます。デフォルトでは、リストは部品番号でソートされます。任意の列見出しをクリックすることで、異なるカテゴリで結果をソートできます。ソート順は昇順から降順へ反転することもできます。

Motorprise には、ゲスト（または非登録）ユーザーは製品の価格情報を表示できないというポリシーがあります。ゲスト・ユーザーが価格を表示せずに製品を比較できるように、/atg/commerce/catalog/comparison/ProductList にあるコンポーネント ProductComparisonList が使用されています。これは、製品を相互に比較するときに表示される製品のプロパティを制御する、TableInfo オブジェクトへの参照を持ちます。デフォルトでは、製品比較リストは、/atg/commerce/catalog/comparison/TableInfo にあるコンポーネントを使用するように構成されます。

Motorprise では、/atg/projects/b2bStore/catalog にある 2 つの TableInfo コンポーネント（MemberComparisonTable および GuestComparisonTable）が使用されています。MemberComparisonTable には price 列が含まれますが、GuestComparisonTable には含まれません。

Motorprise には複数のロケールがあるため、TableInfo.columnHeadings プロパティを使用して、列名をロケールに固有の列見出しにマップするようになっています。表見出しのローカライズの詳細は、『ATG Web Commerce Page Developer's Guide』の「*ソート可能な表の実装*」を参照してください。

MemberComparisonTable.properties

```
$class=atg.service.util.TableInfo
$scope=session

columns=\
     part=sku.manufacturer_part_number,\
     product=productLink ; product.displayName,\
     description=product.description,\
     manufacturer=product.manufacturer.displayName,\
     price=priceInfo.amount,\
     availability=inventoryInfo.availabilityStatusMsg

columnHeadings=\
     part=Part \#,\
     part_de=Produktnr.,\
     product=Product,\
     product_de=Produkt,\
     description=Description,\
     description_de=Beschreibung,\
     manufacturer=Mfr,\
     manufacturer_de=Hrst.,\
     price=Price,\
```

```
     price_de=Preis,\
     availability=Availability,\
     availability_de=Verfügbarkeit
```

GuestComparisonTable.properties

```
$class=atg.service.util.TableInfo
$scope=session

columns=\
     part=sku.manufacturer_part_number,\
     product=productLink ; product.displayName,\
     description=product.description,\
     manufacturer=product.manufacturer.displayName,\
     availability=inventoryInfo.availabilityStatusMsg

columnHeadings=\
     part=Part \#,\
     part_de=Produktnr.,\
     product=Product,\
     product_de=Produkt,\
     description=Description,\
     description_de=Beschreibung,\
     manufacturer=Mfr,\
     manufacturer_de=Hrst.,\
     availability=Availability,\
     availability_de=Verfügbarkeit
```

Product=のエントリでは、列のプロパティ名として式 productLink ; product.displayName が使用されることに注意してください。これにより、表内でプロパティ productLink を表示して、製品の displayName でソートするように、TableInfo コンポーネントに伝えられます。これが行われるのは、productLink が製品のカタログ・ページへのリンクを指定する HTML アンカー・タグであるためです。productLink の典型的な値は、次のような形式です。

```
<dsp:a href="/Motorprise/en/catalog/product.jsp?id=prod70004"/>Wheel
   Bearing</desp:a>
<dsp:a href="/Motorprise/en/catalog/product.jsp?id=prod110145"/>Disk Brake
   Pad Kit</dsp:a>
```

このようなアンカー・タグのリテラル・テキストに基づいてソートした場合、製品 ID の順で Wheel Bearing エントリが Disk Brake Pad Kit より先となります。TableEntry の機能を使用して別個のソート・プロパティを指定すると、リンクのリテラル・テキストにかかわらず、ユーザーが目にする表示名に基づいてこれらの品目がソートされます。

表示されるテーブルを選択するためには、compare_result.jsp 内の次のコードが使用されています。

```
<dsp:importbean bean="/atg/commerce/catalog/comparison/ProductList"/>
<dsp:importbean bean="/atg/userprofiling/Profile"/>
<dsp:importbean bean="/atg/projects/b2bstore/catalog/MemberComparisonTable"/>
<dsp:importbean bean="/atg/projects/b2bstore/catalog/GuestComparisonTable"/>
```

```
<%/*
  First decide which table info object to use when building the product
  comparison table.  Members who are logged in get one table, while guests
  get a different table that omits price information.
*/%>

<dsp:droplet name="Switch">
  <dsp:param bean="Profile.transient" name="value"/>
  <dsp:oparam name="true">
    <dsp:setvalue bean="ProductList.tableInfo" beanvalue="GuestComparisonTable"/>
  </dsp:oparam>
  <dsp:oparam name="false">
    <dsp:setvalue bean="ProductList.tableInfo" beanvalue="MemberComparisonTable"/>
  </dsp:oparam>
</dsp:droplet>
```

このコードでは、ユーザーがログインしているかどうかを確認し、それに基づいて、製品比較リストの `tableInfo` プロパティを、ゲスト・テーブル定義（価格列が含まれない）またはメンバー・テーブル定義（価格列が含まれる）に設定します。ページの残りの部分では `ProductList.tableInfo` を参照し、ユーザーがログインしているかどうかに応じて現在のユーザーに適用される `TableInfo` コンポーネントを取得できます。

通常、Motorprise では、『ATG Web Commerce Page Developer's Guide』の*ソート可能な表の実装*に関する項に記載されているパターンと方法を使用して製品比較表を表示します。ただし、表の `Price` 列には特別なロジックが適用され、ユーザーのロケールにあわせて価格が正確に書式設定され、適切な通貨記号が表示されるようになっています。

対応する `TableInfo` コンポーネントの構成を編集することで、表の列数とその順序を変更できます。価格が表示される列はわからないので、表内の値を反復するときに次のコードを使用して各列見出しをチェックし、列見出しが `Price` の場合は `currency` タグ・コンバータを追加し、他のすべての値に対しては `valueishtml` タグ・コンバータを追加するようになっています。

```
<dsp:droplet name="ForEach">
<%/*
  Here we are going to iterate over all of the property names
  we want to display in this row using BeanProperty to display
  each one.  We handle the "Price" column specially, using a
  currency converter on the value to get properly localized
  currency symbols and formatting.
  */%>
  <dsp:param bean="ProductList.tableColumns" name="array"/>
  <dsp:param name="sortProperties" value=""/>
  <dsp:oparam name="output">
    <td><dsp:droplet name="BeanProperty">
    <dsp:param name="bean" param="currentProduct"/>
    <dsp:param name="propertyName" param="element.property"/>
    <dsp:oparam name="output">
      <dsp:droplet name="Switch">
        <dsp:param name="value" param="element.name"/>
        <dsp:oparam name="price"><dsp:valueof converter="currency"
```

```
          param="propertyValue"/></dsp:oparam>
         <dsp:oparam name="default"><dsp:valueof valueishtml="<%=true%>"
           param="propertyValue"/></dsp:oparam>
      </dsp:droplet>
    </dsp:oparam>
    </dsp:droplet></td>
  </dsp:oparam>
</dsp:droplet>
```

このロジックが正しく動作するためには、JSP で使用する名前が TableInfo コンポーネントで使用する名前と一致している必要があります。たとえば、Motorprise の場合は、TableInfo コンポーネントが編集され、英語およびドイツ語のカタログ・ページの列名が変更されています。そのため、compare_result.jsp で使用する列名もそれにあわせて変更されています。

8 B2B パーソナライズ

Motorprise は、各顧客組織に対してカスタマイズされた操作を提供するように設計されています。特定のカタログと価格表（数量価格設定を含む）を定める契約が各顧客組織に対して作成されています。この章は次の項から構成されています。

カスタム・カタログの作成
Motorprise 顧客の USMW および NDAP 用のカスタム・カタログの作成方法について説明します。

価格表の作成
Motorprise 顧客の USMW および NDAP 用のカスタム価格表の設定方法について説明します。

数量価格設定の使用
ATG Control Center を使用して一括価格設定および段階的価格設定を行う方法について説明します。

契約の作成
USMW および NDAP との契約の設定方法について説明します。

カスタム・カタログの作成

Motorprise では、USMW および NDAP のユーザー向けにカタログがカスタマイズされており、それぞれのビジネスに関連するカテゴリと製品のみが表示されます。

Master Catalogs、Subcatalogs および Customer Catalogs のフォルダ構造が設定されています。また、Subcatalog フォルダ内に、すべてのサブカタログ（メインまたはルート・カテゴリごとに 1 つ）が作成されています。たとえば、Brakes というサブカタログがあります。このメイン・カテゴリ内に、ブレーキ・ラインやドラムなど、ブレーキ関連のすべての製品が置かれます。

カタログがこのように設定された理由は次のとおりです。

- 特定の製品グループを特定の顧客に簡単に表示します。
- 顧客間でカテゴリを共有できるため、顧客ごとに個別のカタログを作成する必要がありません。
- 同じカテゴリを使用して、マスター・カタログを作成します。
- Motorprise 製品のスペシャリストがカテゴリを管理できます。

これらのビジネス・ニーズは Motorprise 固有のものです。Oracle Commerce Core Commerce の柔軟なカタログ管理機能を使用して、各自のビジネス戦略に特に適したカタログ構造を作成および管理できます。

各サブカタログ内には、カタログ構造を構成するカテゴリが作成されています。各カテゴリには、複数のサブカテゴリがあります。製品と SKU はこれらのサブカタログ内に作成され、そこに置かれます。

たとえば、Electrical サブカタログには Bulbs というカテゴリがあり、このカテゴリにはそのすべてが bulb という名前を持つ複数の製品が含まれています。これらの各製品について、そのプロパティと画像の管理、クロス

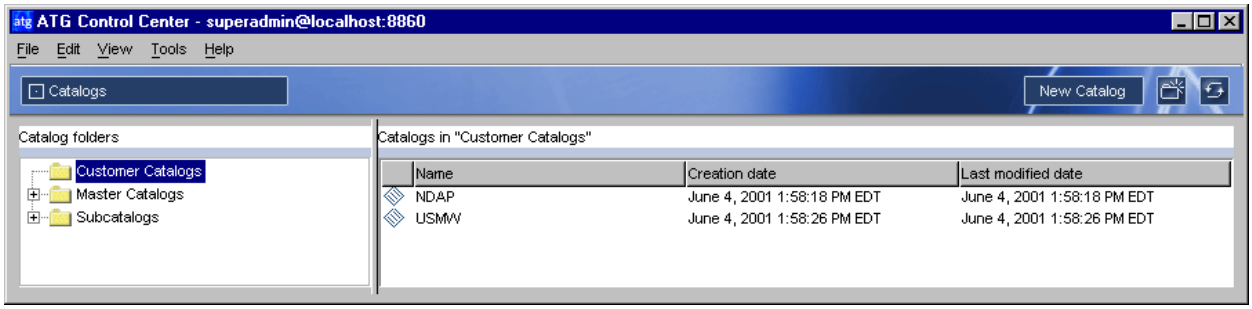
セル関係の定義、製品の変動または SKU の管理、およびこの特定の製品が含まれる他のカタログの表示も行うことができます。

（カタログはカテゴリを共有できません。しかし、他のカタログを共有することはできます。このため、USMW と NDAP は、Electrical など、サブカタログ内のカテゴリを共有できます。）

また、Master Catalog フォルダ内にベース・カタログが作成され、それにすべてのサブカタログが追加されています。ベース・カタログ内のすべてのカテゴリと製品を表示し、**「ここをクリック」**リンクを使用してサブカタログに移動し、それを編集できます。

匿名ユーザーなど、カタログのない Motorprise サイト・ユーザーには、デフォルトでベース・カタログが表示されます。この機能の詳細は、このマニュアルの「マーチャンダイジング」の Catalog シナリオに関する項を参照してください。

さらに、USMW および NDAP の顧客カタログが作成されています。USMW には、Electrical、Cooling System、Brakes、Ignition and Tune-Up、Exhaust および Air and Fuel というカテゴリがあります。NDAP には、Belts and Hoses、Electrical、Cooling System、Brakes、Ignition and Tune-Up、Tools および Air and Fuel というカテゴリがあります。USMW および NDAP の顧客カタログは様々なサブカタログで構築されており、その中にすべてのカテゴリ、製品および SKU が含まれています。使用するサブカタログは、それぞれのビジネス要件に基づいて決定されています。たとえば、USMW は Motorprise から工具を購入しないため、Tools カテゴリは表示されません。



Motorprise 内のカタログ

ATG Control Center で、このような顧客カタログ内のすべての製品を表示できます。また、任意のカタログ内で**「ここをクリック」**リンクを通じて製品にアクセスし、編集することもできます。

詳細は、『ATG Web Commerce Programming Guide』を参照してください。

カタログのローカライズ

Motorprise は、次の 2 種類の言語をサポートするように設計されています: 英語とドイツ語。サイトのコンテンツはこの 2 種類の言語で使用できます。このローカライゼーションは、カタログにドイツ語の属性を追加することによって実現されました。

これを完成させるために、導出プロパティおよび `FirstWithLocale` という導出メソッドが使用されました。

例として、Motorprise カタログ `<ATG11dir>/Motorprise/config/atg/commerce/catalog/custom/customCatalog.xml` 内の `category.displayName` を使用しましょう。これは、「Brakes」や「Tools」など、カタログ・ページに表示されるカテゴリの名前です。

最初に、デフォルトのカタログ・スキーマから、`displayName` の既存の定義が削除されました。

```
<table name="dcs_category" type="primary" id-column-name="category_id">
  <property name="displayName" xml-combine="remove" />
```

remove コマンドは、正しい table タグ内に指定する必要があります。正しいタグ内に指定されないと、displayName の定義が適切に削除されません。このコマンドは DCS テーブルを上書きします。

これを削除するのは、この汎用プロパティを置換するためであり、英語の表示名およびドイツ語の表示名に置換します。

次に、同じ table タグ内で、英語の表示名の定義が作成されました。同じデータベース内の同じ列を再利用したため、データの移行を行う必要はありませんでした。displayName の英語値はこの列にすでに存在します。この新規プロパティに displayName_en という名前を付け、attribute タグを使用して、値が en の locale を割り当てます。

```
<property name="displayName_en" data-type="string" column-name=
   "display_name" required="true" queryable="true" category-
   resource="categoryPresentation" display-name-resource="displayName">
 <attribute name="propertySortPriority" value="-8" />
 <attribute name="locale" value="en" />
</property>
```

displayName に対して使用したものと同じ category-resource および display-name-resource を、displayName_en に対して使用したことに注意してください。この後、displayName_de に対しても同じものを使用したことに気付かれるでしょう。ATG Control Center は、プロパティが locale 属性を持つ場合を認識できます。この属性はリソース文字列の末尾に追加されます。たとえば、リソース文字列が「Display name」の場合、英語版は「Display name（en）」になります。このプロパティ定義の最も重要な部分は、locale 属性です。必要に応じて、en_US など、より具体的な値を使用できます。次に記載されている導出メソッドではどちらも認識します。

英語の表示名は作成されましたが、Motorprise ではドイツ語の表示名も必要です。

最初に、新規テーブル dbc_category_de をデータベース内に作成し、カテゴリのドイツ語プロパティをすべて格納しました（keywords などの複数値プロパティは除きます）。

新規 table タグ内に、ドイツ語の表示名 displayName_de を作成します。

```
<table name="dbc_category_de" type="auxiliary" id-column-name=
   "category_id">
   <property name="displayName_de" data-type="string" column-name=
      "display_name" queryable="true" category-resource=
      "categoryPresentation" display-name-resource="displayName">
      <attribute name="propertySortPriority" value="-8" />
      <attribute name="locale" value="de" />
  </property>
</table>
```

これで、英語の表示名とドイツ語の表示名が作成されました。ユーザーがサイトにアクセスしたときに、ユーザーのロケールに基づいて正しい言語が選択される必要があります。これを行うには、displayName プロパティを再作成し、FirstWithLocale 導出メソッドを使用します（同じ名前を使用したため、JSP の変更は必要ありません）。

```
<property name="displayName" data-type="string" category-resource=
   "categoryPresentationDerived" writable="false" display-name-resource=
   "displayName" queryable="true">
  <derivation user-method="atg.commerce.dp.FirstWithLocale">
 <expression>displayName_en</expression>
 <expression>displayName_de</expression>
 </derivation>
 <attribute name="resourceBundle"
     value="atg.projects.b2bstore.CustomCatalogTemplateResources" />
 <attribute name="propertySortPriority" value="-8" />
 <attribute name="keyService" value="/atg/userprofiling/LocaleService" />
 <attribute name="defaultKey" value="en_US" />
 </property>
```

このプロパティはデータベースに格納されていないため、table タグ内にはありません。keyService は、現在のユーザーのロケールを指示する Nucleus コンポーネントです。ロケールの値が KeyService によって提供されない場合は、defaultKey が使用されます。たとえば、ユーザーのロケールが sn_ZW の場合、ジンバブエのショナ語用の表示名は定義されていないため、英語の表示名が表示されます。導出メソッドは最初、KeyService によって返される完全なロケール（de_DE など）を検索します。何も見つからない場合は、次にロケールの言語部分のみを検索します。displayName_de のロケールを de に設定するのはそのためであり、ロケールが de_DE の場合も引き続き使用されます。

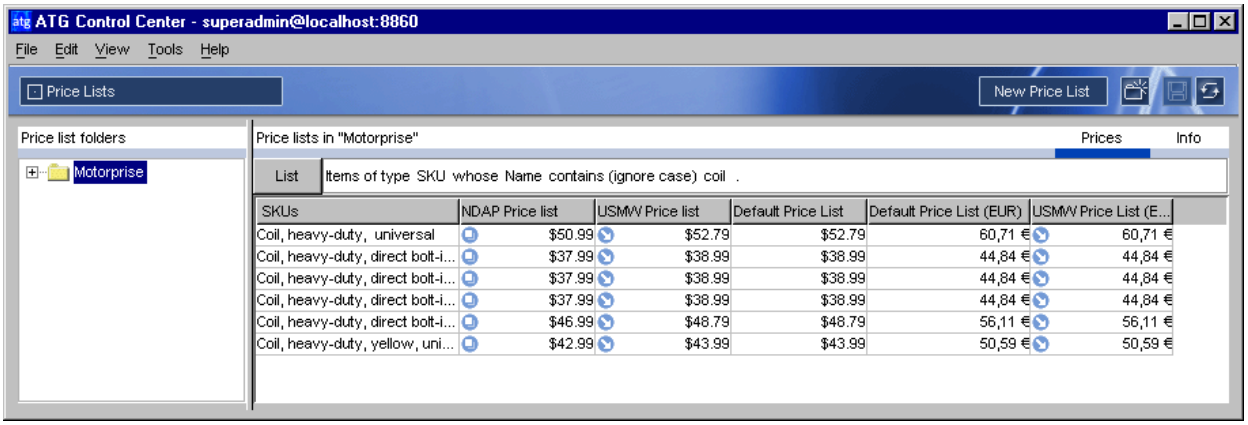
category のドイツ語プロパティの定義を、次に示します。keywords 以外のすべてが 1 つの table 内にあることに注目してください。

```
<!-- create german versions in a separate table -->
<table name="dbc_category_de" type="auxiliary" id-column-name=
   "category_id">

  <property name="displayName_de" data-type="string" column-name=
   "display_name" queryable="true" category-resource=
   "categoryPresentation" display-name-resource="displayName">
    <attribute name="propertySortPriority" value="-8"/>
    <attribute name="locale" value="de"/>
  </property>

  <property name="description_de" data-type="string" column-name=
     "description" queryable="true" category-resource=
     "categoryPresentation" display-name-resource="description">
    <attribute name="propertySortPriority" value="-7"/>
    <attribute name="locale" value="de"/>
  </property>

  <property name="longDescription_de" data-type="string" column-
     name="long_description" queryable="true" category-resource=
     "categoryPresentation" display-name-resource="longDescription">
    <attribute name="propertySortPriority" value="-6"/>
    <attribute name="locale" value="de"/>
  </property>
```

```
  <property name="template_de" item-type="media" column-name="template_id"
     queryable="true" category-resource="categoryPresentation"
     display-name-resource="template">
    <attribute name="propertySortPriority" value="-5"/>
    <attribute name="locale" value="de"/>
  </property>

</table>

<!-- create german keywords -->
<table name="dbc_cat_key_de" type="multi" id-column-name="category_id"
   multi-column-name="sequence_num">

  <property name="keywords_de" data-type="list" component-data-type=
     "string" column-name="keyword" category-resource=
     "categoryPresentation" display-name-resource="keywords">
     <attribute name="propertySortPriority" value="-4"/>
     <attribute name="locale" value="de"/>
  </property>
</table>
```

価格表の作成

Motorprise では、顧客との特別価格の交渉がしばしば行われます。この特別価格を管理するために、Motorprise には様々な価格表が作成されています。別の顧客の価格表に他に価格がない場合は、`defaultPriceList` が使用されます。USMW と NDAP の価格を管理するために、USMW 用と NDAP 用のカスタム価格表も設定されています。たとえば、ATG Control Center の「価格設定」→「価格表」領域で、名前に Coil が含まれる SKU を検索すると、カスタマイズされた価格設定が表示されます。



Motorprise は USMW と NDAP に対してカスタマイズされた価格設定を提供する。

詳細は、『ATG Web Commerce Programming Guide』および『ATG Web Commerce ストア設定コマース・ガイド』を参照してください。

価格表のローカライズ

USMW Price List（Eur）という名前の、USMWドイツ部署で使用される価格表も作成されています。これは defaultPriceList と同一ですが、ユーロに変換されたものです。どちらの価格表も、locale プロパティを持ちます。USMWドイツ価格の場合、価格はユーロで定義されるため、このプロパティは de_DE_EURO に設定されます。USドル価格の場合は、ロケールのデフォルト通貨が USドルのため、locale は en_US です。各ユーザーのプロファイルに正しい価格表を割り当てる必要があります。

ユーザーの価格表は、親組織の契約によって定義されます。このため、登録ユーザーがサイトの表示言語を変更しても、同じ価格が表示されます。

CurrencyFormatter、CurrencyConversionTagConverter および EuroTagConverter の各プロパティを組み合せた、新規クラス atg.droplet.CurrencyConversionFormatter が作成されています。

USドルとユーロの価格表を変換および書式設定するために、このドロップレットが使用されています。これは、変換する金額、元のロケール（価格表のロケール）および変換先のロケールを入力として受け取ります。

出力パラメータ（oparam="output"内）は 1 つで、それは formattedCurrency です。

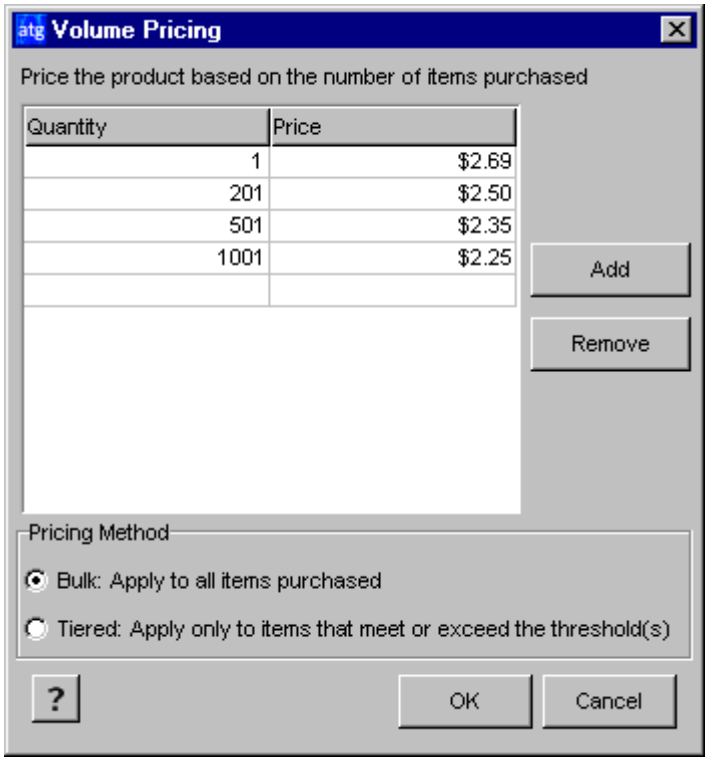
```
<droplet bean="/atg/dynamo/droplet/CurrencyConversionFormatter">
  <param name="currency" value="param:price.listPrice">
  <param name="locale" value="en_US">
  <param name="targetLocale" value="de_DE_EURO">
  <param name="euroSymbol" value="&euro;">
  <oparam name="output">
    <valueof param="formattedCurrency" valueishtml>no price</valueof>
  </oparam>
</droplet>
```

CurrencyConversionFormatter の詳細は、『ATG Web Commerce Programming Guide』を参照してください。

数量価格設定の指定

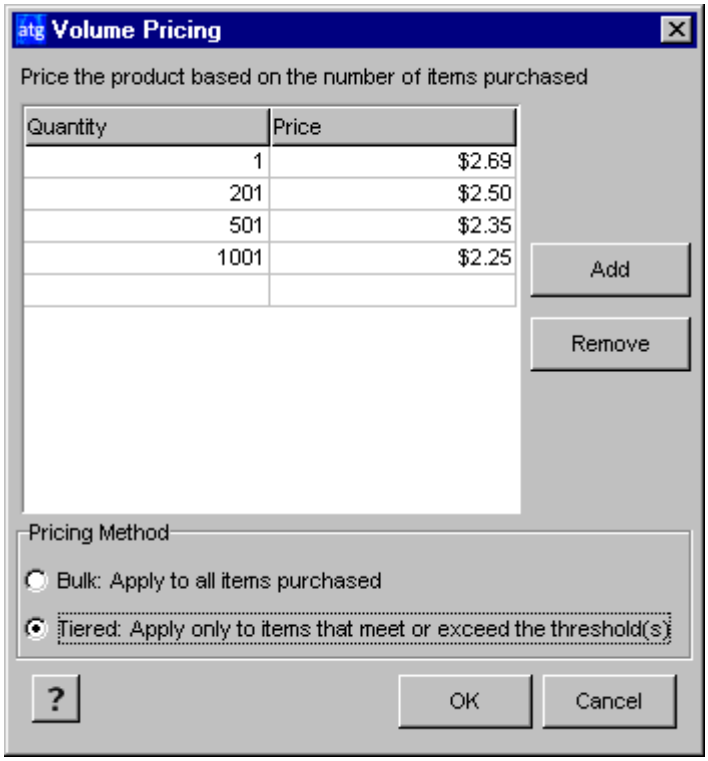
Motorprise では数量価格設定が使用されています。Core Commerce には、2 種類の数量価格設定があります。

一括: 購入するすべての品目に価格が適用されます。たとえば、次の画面で 500 個を購入した場合、1 つ当たりの値段は$2.50 です。



ATG Control Center における一括価格設定

段階的: しきい値以上の品目にのみ価格が適用されます。たとえば、次の画面で 500 個を購入した場合、最初の 1～200 個については 1 つ当たりの値段が$2.69、次の 300 個については 1 つ当たりの値段が$2.50 です。



ATG Control Center における段階的価格設定

特定の品目に数量価格設定を割り当てるために、ATG Control Center が使用されています。ボリューム価格設定の詳細は、『ATG Web Commerce Programming Guide』を参照してください。

契約の作成

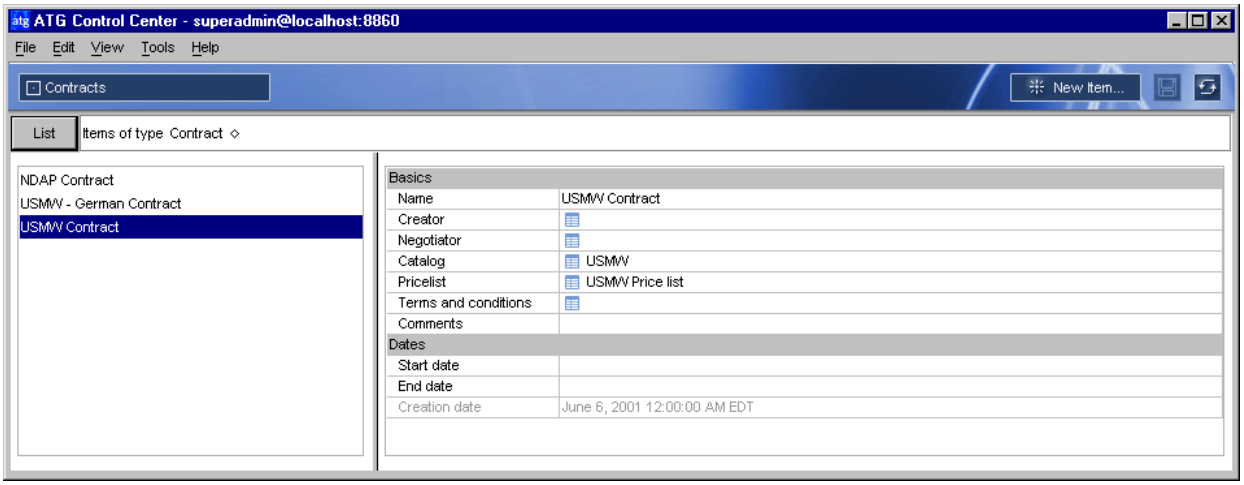
Motorprise では、契約を通じてカタログが組織に割り当てられます。ATG Control Center の「購買および支払」→「契約」領域で、契約を作成できます。価格表およびカタログを組織に関連付けるために、契約が使用されています。

Motorprise では、各組織の契約は 1 つのみです。しかし、1 つの組織で複数の契約を使用し、ユーザーがログインしたときにデフォルトを現在アクティブな契約に設定するように、サイトを簡単に拡張できます。

各組織は `Contract` プロパティを持ち、これが契約から `Catalog` プロパティおよび `Pricelist` プロパティを導出します。

ユーザーは `Contract` プロパティを継承し、したがってその `Catalog` プロパティおよび `Pricelist` プロパティも継承されます。契約を使用せずに、カタログと価格表を組織に直接関連付けることができます。その場合、ユーザーは `Catalog` プロパティおよび `Pricelist` プロパティを組織から直接継承します。

詳細は、『ATG Web Commerce Programming Guide』を参照してください。



USMW 契約によってUSMW に価格表およびカタログが関連付けられる。

契約のローカライズ

同じ製品カタログを使用してドイツ USMW に固有の契約が作成され、それに対して異なる価格表が割り当てられています。

ドイツ Motorprise ユーザーには、製品ページと販促の価格がユーロとドイツ・マルクの両方で表示されます。しかし、精算、オーダー履歴および比較の各ページにはユーロのみが表示されます。これらのページでは、契約で定義される実際の通貨で価格が表示されます。

9 オーダーの処理

この章では、Motorprise サイトのオーダー取得機能について説明します。この章は次の項から構成されています。

現在のオーダー
オーダーを編集および保存する方法について説明します。

出荷情報
デフォルト所在地、代替所在地、所在地間での品目の分割、複数の出荷方法など、Motorprise の出荷機能について説明します。

支払情報
P.O.番号、購買依頼、コスト・センターなど、Motorprise の請求機能について説明します。

コスト・センターの指定
ユーザーがコスト・センターを指定する方法について説明します。

オーダーの確認および確定
オーダーを確認および発行する方法について説明します。

Motorprise ユーザーは、オーダーの発行時に多くのオプションを使用できます。異なる出荷方法、請求方法および所在地の中から選択できます。明細品目または金額でオーダーを分割することもできます。

現在のオーダー

Motorprise では、「Current Order」ページ（`checkout/cart.jsp`）にオーダーの各明細品目、数量と価格、および現在の小計が表示されます。ユーザーは品目の数量変更、品目の削除、オーダーの保存、または精算プロセスに進むことができます。

オーダーを保存および更新するユーティリティを提供するために、`checkout/cart.jsp` で様々なフォーム・ハンドラが使用されています。

オーダーの保存

ユーザーは現在のオーダーに品目を保存し、後でオーダーを発行することができます。「**Save Order**」ボタンによってユーザーがリダイレクトされる `user/save_order.jsp` では、`/atg/commerce/order/purchase/SaveOrderFormHandler` を使用して指定のオーダーを保存します。

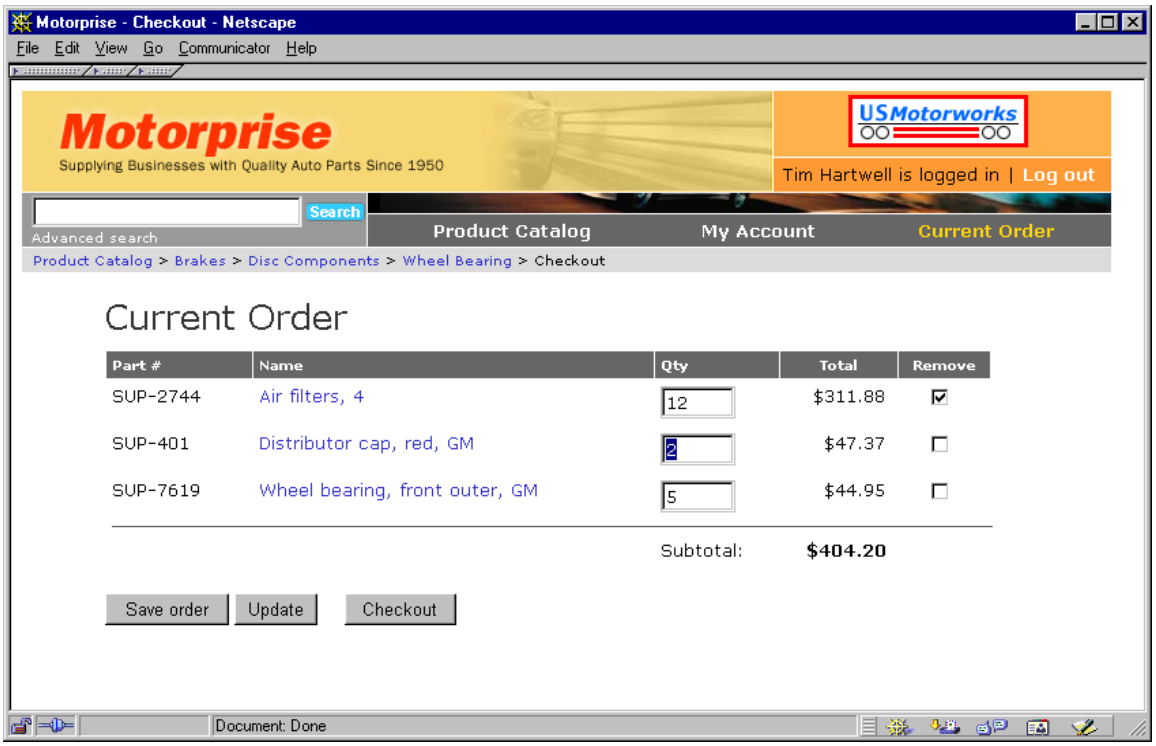
```
<dsp:input bean="SaveOrderFormHandler.description" type="text"/>
<dsp:input bean="SaveOrderFormHandler.saveOrder" type="submit" value="Save
   order"/>
<dsp:input bean="SaveOrderFormHandler.saveOrder" type="hidden" value="save"/>
<dsp:input bean="SaveOrderFormHandler.saveOrderSuccessURL" type="hidden"
```

```
   value="../user/saved_orders.jsp"/>
<dsp:input bean="SaveOrderFormHandler.saveOrderErrorURL" type="hidden"
   value="../user/save_order.jsp"/>
```

オーダーの保存の詳細は、このマニュアルの「自分のアカウント」を参照してください。

オーダーの更新

「Current Order」ページでは、オーダーから品目を削除することや、品目の数量を変更することができます。数量を変更するには「Qty」フィールドに新規数量を入力し、明細品目を削除するには「Remove」チェック・ボックスを選択します。このような変更を行った後、「**Update**」ボタンをクリックします。



*数量を変更したり明細品目を削除してから「**Update**」ボタンをクリックする。*

`checkout/cart.jsp` は、`ForEach` ドロップレットを使用して、オーダー内の各明細品目を反復するようになっています。明細品目ごとに表示される出力には、現在の数量、価格、参照製品ページへのリンク、数量を変更したり品目を削除する機能など、重要な情報が含まれています。

オーダーを更新するためには、`checkout/cart.jsp` 内で `/atg/commerce/order/purchase/CartModifierFormHandler.setOrder` ハンドラ・メソッドが使用されています。このハンドラ・メソッドは各 `CommerceItem` の発行数量を判断して、数量が変更されたものについては調整し、数量がゼロ以下のものは削除します。

品目の数量の変更

オーダー内の品目の数量を変更するには、CartModifierFormHandler に、CommerceItem.catalogRefId という名前で、その値は CommerceItem の新規数量に等しい要求パラメータを渡します。

```
<input type="text" size="3"
  name="<dsp:valueof param='CommerceItem.catalogRefId'/>"
  value="<dsp:valueof param='CommerceItem.quantity'/>">
```

品目の削除

オーダーから品目を選択的に削除するには、CartModifierFormHandler.removalCatalogRefIds プロパティを catalogRefIds の配列に設定します。

```
<%-- Display "remove" checkbox column --%>
<td align=middle>
   <dsp:getvalueof id="skuId" param="CommerceItem.catalogRefId">
      <dsp:input type="checkbox" bean=
         "CartModifierFormHandler.removalCatalogRefIds" value="<%=skuId%>"/>
   </dsp:getvalueof>
</td>
```

オーダーの更新

数量を変更し、削除する品目をマークしたら、「Update」をクリックしてショッピング・カート・ページを更新するか、「Checkout」をクリックして精算プロセスに進みます。どちらの発行ボタンも、要求内の変更に基づいてオーダーを変更するハンドラを起動します。また、「Checkout」ボタンによってユーザーは checkout/shipping.jsp にリダイレクトされます。次のコード・スニペットは、Update ハンドラおよび Checkout ハンドラを示しています。

```
<%-- Update Order button: --%>
<dsp:input bean="CartModifierFormHandler.setOrder" type="submit"
value="Update"/>

<%--
GoTo this URL if user pushes RECALCULATE button and there are no errors:
--%>
<dsp:input bean="CartModifierFormHandler.setOrderSuccessURL" type="hidden"
   value="../checkout/cart.jsp"/> <%/* stay here */%>

<%--
GoTo this URL if user pushes RECALCULATE button and there are errors:
--%>
<dsp:input bean="CartModifierFormHandler.setOrderErrorURL" type="hidden"
   value="../checkout/cart.jsp"/> <%/* stay here */%>

<%-- CHECKOUT Order button: --%>
      <dsp:input bean="CartModifierFormHandler.moveToPurchaseInfo"
   type="submit" value="Checkout"/>
```

```
<dsp:input bean="CartModifierFormHandler.moveToPurchaseInfoSuccessURL"
   type="hidden" value="../checkout/shipping.jsp"/> <%-- move on to shipping --%>
```

詳細は、『ATG Web Commerce Programming Guide』を参照してください。

出荷情報

各 Motorprise ユーザーには、デフォルト出荷先所在地、および許可された出荷先所在地の代替リストがあります。これらの所在地は、プロファイルと組織のプロパティの defaultShippingAddress および shippingAddrs で保持されます。Motorprise では、これらは親組織から継承され、その組織の管理者によってのみ変更できます。他のバイヤーが新規出荷先所在地を入力することや、既存の出荷先所在地を変更することはできません。出荷先所在地を編集する機能はそれぞれのサイトで異なる方法で構成できます。

Motorprise では、ユーザーは各自の複雑なビジネス・ニーズに合った様々な出荷オプションを選択できます。次の操作を行うことができます。

- オーダー内のすべての品目をデフォルト出荷先所在地に出荷します。
- 代替のリストから異なる所在地を選択します。
- オーダー内の明細品目を様々な出荷先所在地に分割します。
- 所在地ごとに 2 日便出荷、翌日便出荷など、様々な出荷方法を選択します。

Motorprise 製品はすべて、物理的な所在地が必要な商品です。これらの所在地は、HardgoodShippingGroup オブジェクトの形式でオーダーに関連付けられます。プロファイル内の所在地に基づいて HardgoodShippingGroups を作成するためには、checkout/shipping.jsp で /atg/commerce/order/purchase/ShippingGroupDroplet が使用されています。このコンポーネントによって、ユーザーの defaultShippingAddress プロファイル・プロパティに基づいてデフォルトの HardgoodShippingGroup も作成されます。

これらの ShippingGroups は、購入プロセス時にユーザーの出荷グループを格納するために使用される ShippingGroupMapContainer に追加されます。このコンテナは Nucleus によってインスタンス化され、/atg/commerce/order/purchase/ShippingGroupContainerService に置かれます。

さらに、オーダー内の明細品目ごとに、ShippingGroupDroplet は CommerceItemShippingInfo オブジェクトを作成し、それを CommerceItemShippingInfoContainer に追加します。CommerceItemShippingInfo は、オーダーの明細品目、および出荷情報を表す HardgoodShippingGroups の間の関係を追跡するヘルパー・オブジェクトです。各 ShippingGroup に属する各品目の数量は、これを使用して追跡されます。購入プロセス時にユーザーのヘルパー・オブジェクトを格納するには、コンテナが使用されます。

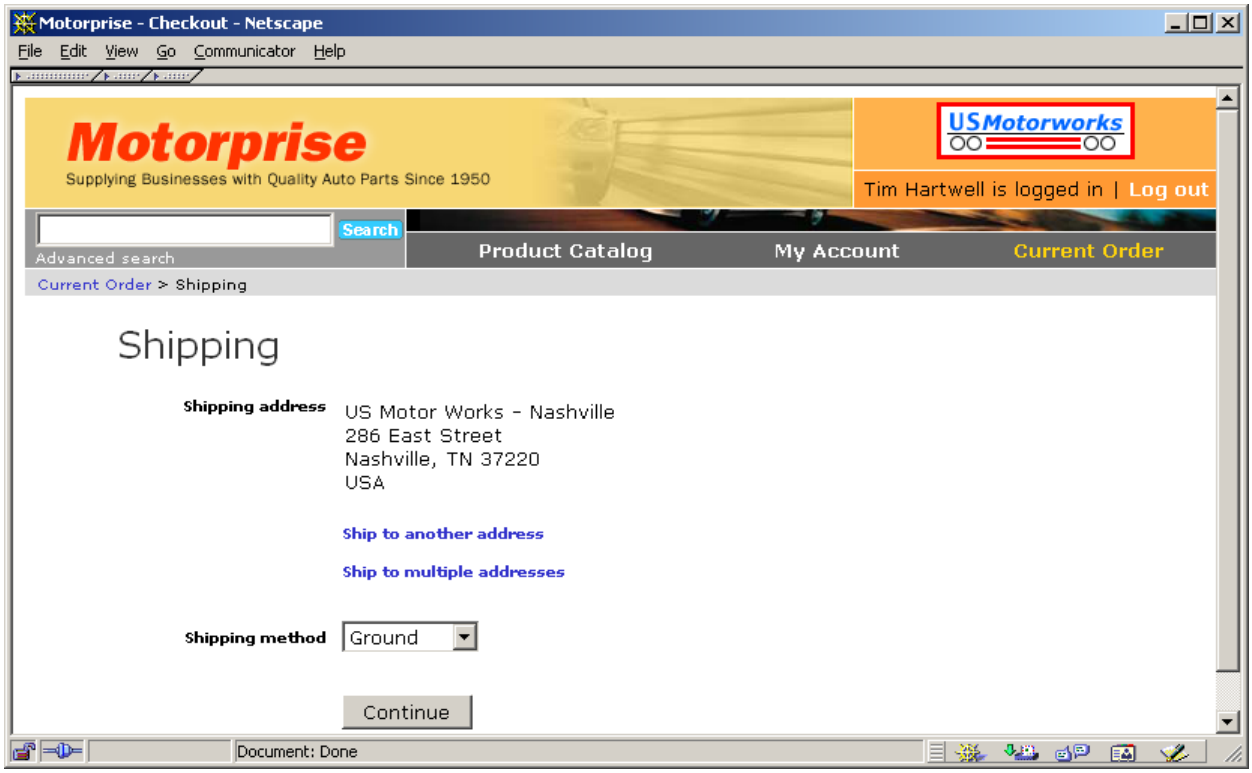
ShippingGroupMapContainer の詳細は、『ATG Web Commerce Platform API Reference』のクラス atg.commerce.order.purchase.ShippingGroupContainerService に関する説明を参照してください。

checkout/shipping.jsp を調べて、ShippingGroupDroplet の使用方法を確認できます。詳細は、『ATG Web Commerce Programming Guide』を参照してください。

デフォルト所在地の選択

オーダー内のすべての品目がデフォルト出荷先所在地に出荷されるように選択できます。この場合は、出荷方法を選択し、「**Continue**」ボタンをクリックするだけです。この操作によって、`ShippingGroupFormHandler.applyShippingGroups` メソッドが起動されます。標準の `validateShippingInfo` パイプライン・チェーンが実行され、ユーザーは `checkout/billing.jsp` にリダイレクトされます。

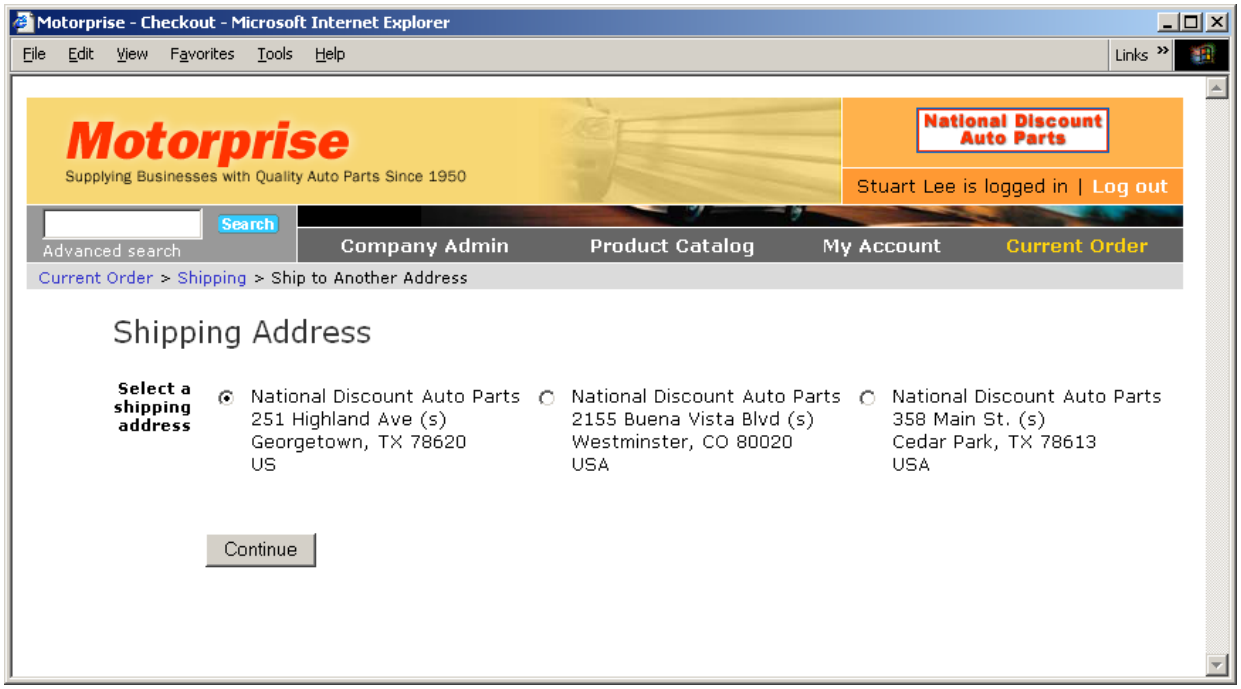
`validateShippingInfo` パイプライン・チェーンの詳細は、『ATG Web Commerce Programming Guide』を参照してください。



デフォルト出荷先所在地の使用

代替所在地への出荷

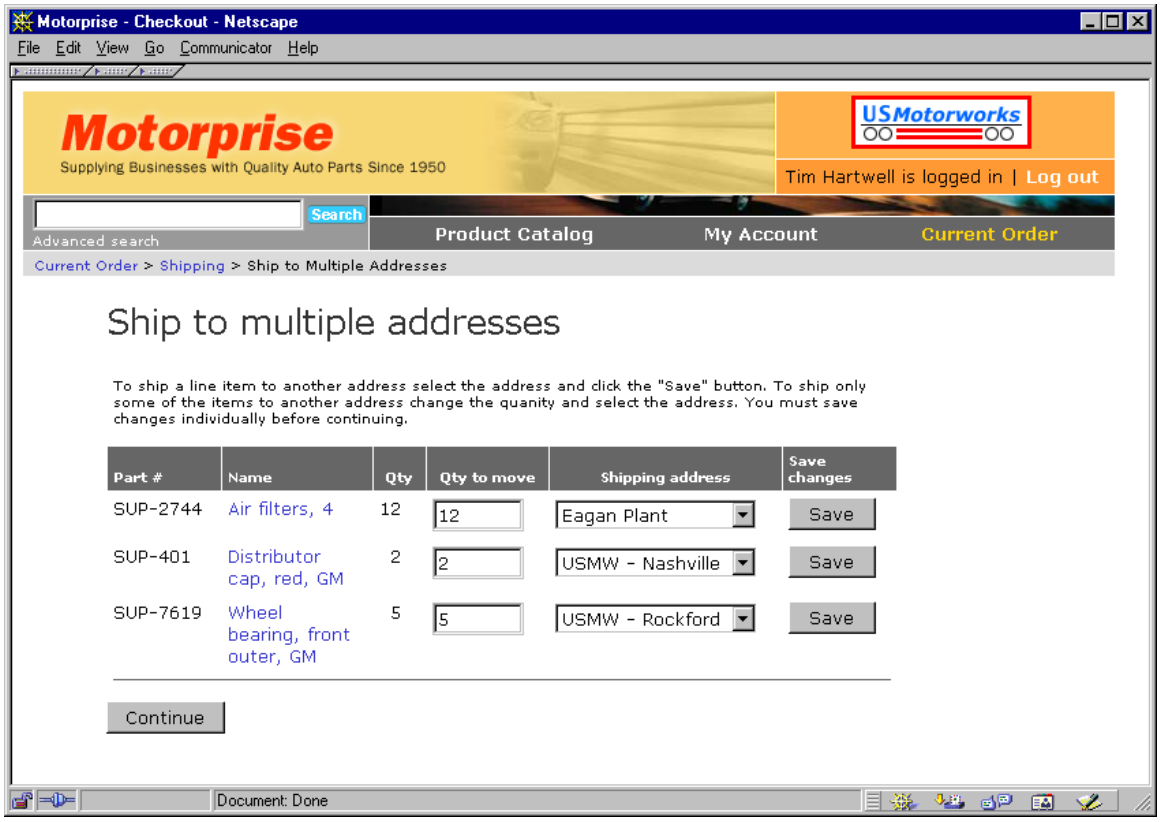
「**Ship to another address**」リンクをクリックして、すべての品目を代替出荷先所在地に出荷することもできます。ユーザーは `checkout/shipping_address.jsp` にリダイレクトされ、使用可能なすべての所在地が表示されます。使用可能なすべての出荷先所在地から任意の所在地を選択し、この所在地における出荷方法を引き続き選択できます。



複数の所在地への出荷

checkout/ship_to_multiple.jsp への「**Ship to multiple addresses**」リンクをクリックすると、オーダー内の品目の複数の所在地への出荷を決定できます。

使用可能な出荷先所在地のドロップダウン・リストが表示されます。目的の所在地および各明細品目の適切な数量を選択し、「**Save**」ボタンをクリックします。



複数の出荷先所在地の割当て

ユーザーの選択を追跡するには、`atg/commerce/order/purchase/ShippingGroupFormHandler` コンポーネントを使用します。これは、様々な明細品目と出荷先所在地への分配方法の間の関連付けのリストを保持しており、これによってユーザーは、ある品目セットを 1 つの所在地に出荷し、別の品目セットを別の所在地に出荷することができます。このコンポーネントによって、異なる数量の同じ品目を異なる所在地に出荷することもできます。

最初に、`ShippingGroupDroplet` が初期化されます。ページは、以前の `ShippingGroupMapContainer` および `CommerceItemShippingInfoContainer` を消去し、再初期化する必要があるかどうかを指定する、`init` という初期化パラメータを受け取ります。ページに初めてアクセスする場合、このパラメータは `true` に設定されます。ユーザーが情報を変更してページがリフレッシュされると、パラメータは `false` に設定されて、ユーザーの変更を維持します。

初期化されると、`ShippingGroupMapContainer` にはユーザーの `ShippingGroups` が格納され、`CommerceItemShippingInfoContainer` には、オーダー内の明細品目ごとに 1 つの `CommerceItemShippingInfo` が格納されます。次の `ship_to_multiple.jsp` からの JSP が、この初期化を行います。

```
<dsp:droplet name="ShippingGroupDroplet">
  <dsp:param name="clearShippingGroups" param="init"/>
  <dsp:param name="initShippingGroups" param="init"/>
  <dsp:param name="shippingGroupTypes" value="hardgoodShippingGroup"/>
```

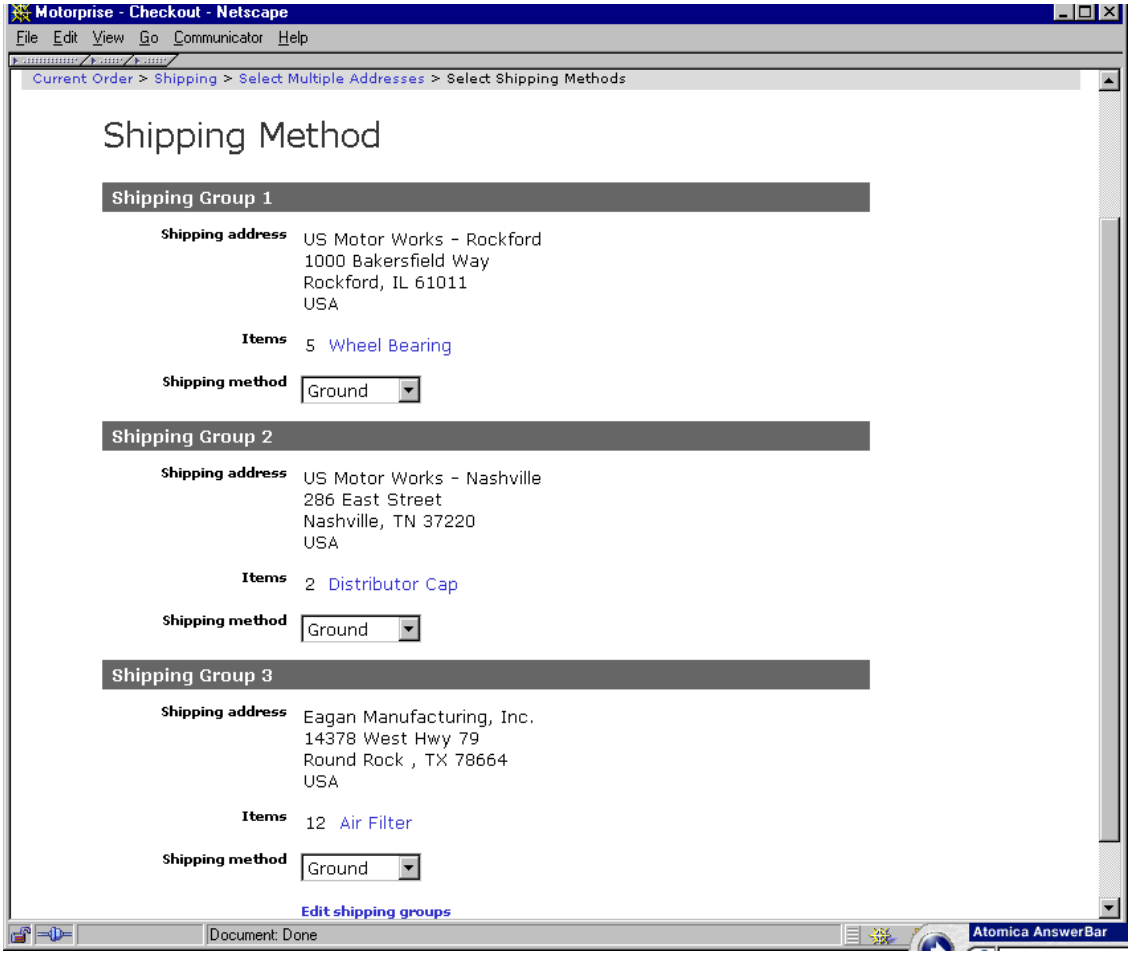
shippingGroupTypes 値に複数のカンマ区切りのエントリを含めることによって（value="hardgoodShippingGroup,electronicShippingGroup"など）、複数の出荷タイプを持つことができます。

ユーザーには、オーダー内の品目ごとに出荷先所在地を指定するフォームが表示されます。デフォルト所在地が自動的に選択されます。ユーザーが行う変更は、基本的に、各 CommerceItem オブジェクトの CommerceItemShippingInfo オブジェクトを変更します。

CommerceItemShippingInfo オブジェクトには、ShippingGroupMapContainer 内の ShippingGroup を、金額で CommerceItem に関連付ける情報が格納されます。CommerceItem を複数の ShippingGroup に関連付ける場合は、CommerceItemShippingInfo オブジェクトがこれらの各関連付けを記述する必要があります。CommerceItemShippingInfoContainer は、これらの関連付けを、CommerceItem をキーとする CommerceItemShippingInfo オブジェクトのリストとして階層的に保持します。

フォームは、オーダー内の CommerceItems を反復することにより作成されています。CommerceItem ごとに、その CommerceItemShippingInfo オブジェクトのリストを反復します。最初、各 CommerceItemShippingInfo オブジェクトは、CommerceItem、その数量、および ShippingGroupMapContainer の DefaultShippingGroupName で指定される ShippingGroup を参照します。

ユーザーはこのフォームで、CommerceItemShippingInfo オブジェクトの SplitQuantity および SplitShippingGroupName プロパティ値を更新し、ShippingGroupFormHandler.splitShippingInfos をコールすることによってこれらを発行できます。このように、CommerceItems は、ShippingGroupDroplet の初期化で提供されるものではなく、追加のまたは異なる ShippingGroups に関連付けることができます。



複数の出荷グループ

このフォームを作成する checkout/ship_to_multiple.jsp からの JSP を、次に示します。

```
<%--
    For each CommerceItem in the Order, we obtain a List of
    CommerceItemShippingInfo objects.
    Each CommerceItemShippingInfo object associates the CommerceItem to a
    particular ShippingGroup.
--%>

<dsp:droplet name="ForEach">
  <dsp:param name="array" param="order.commerceItems"/>
  <dsp:oparam name="output">
    <dsp:setvalue paramvalue="element" param="commerceItem"/>
    <dsp:setvalue bean="ShippingGroupFormHandler.listId"
      paramvalue="commerceItem.id"/>
    <dsp:droplet name="ForEach">
      <dsp:param bean="ShippingGroupFormHandler.currentList" name="array"/>
```

```
<dsp:oparam name="output">
  <dsp:setvalue paramvalue="element" param="cisiItem"/>
  <dsp:form action="ship_to_multiple.jsp" method="post">
  <tr valign=top>
   <td><nobr><dsp:valueof param=
     "commerceItem.auxiliaryData.catalogRef.manufacturer_part_number"
      /></nobr></td>
   <td></td>
   <td><dsp:a href="../catalog/product.jsp?navAction=jump">
         <dsp:param name="id" param="commerceItem.auxiliaryData.productId"/>
         <dsp:valueof param=
          "commerceItem.auxiliaryData.catalogRef.displayName"/></dsp:a></td>
   <td></td>

   <td align=right><dsp:valueof param="element.quantity"/></td>
   <td> </td>
   <td>

  <%--
            These form elements permit the user to assign ShippingGroups by
            name and for a specific quantity to a CommerceItem.
  --%>

   <dsp:input bean=
     "ShippingGroupFormHandler.currentList[param:index].splitQuantity"
       paramvalue="element.quantity" size="4" type="text"/></td>
   <td> </td>
   <td>
     <dsp:select bean=
       "ShippingGroupFormHandler.currentList[param:index].
       splitShippingGroupName">
     <dsp:droplet name="ForEach">
       <dsp:param name="array" param="shippingGroups"/>
       <dsp:oparam name="output">

         <dsp:droplet name="Switch">
           <dsp:param name="value" param="key"/>
           <dsp:getvalueof id="SGName" idtype="String"
             param="cisiItem.shippingGroupName">
           <dsp:getvalueof id="keyname" idtype="String" param="key">
           <dsp:oparam name="<%=SGName%>">
             <dsp:option selected="<%=true%>" value=
               "<%=keyname%>"/><dsp:valueof param="key"/>
           </dsp:oparam>
           <dsp:oparam name="default">
             <dsp:option selected="<%=false%>" value=
               "<%=keyname%>"/><dsp:valueof param="key"/>
           </dsp:oparam>
           </dsp:getvalueof>
```

```
                  </dsp:getvalueof>

                </dsp:droplet>

                <%-- equivalent code to switch droplet using core:case. Both of
                  these work correctly.
                <dsp:getvalueof id="keyname" idtype="String" param="key">
                <dsp:getvalueof id="SGName" idtype="String"
                  param="cisiItem.shippingGroupName">
                  <core:switch value="<%=keyname%>">
                    <core:case value="<%=SGName %>">
                      <dsp:option selected="<%=true%>" value=
                        "<%=keyname%>"/><dsp:valueof param="key"/>
                    </core:case>


                    <core:defaultCase>
                      <dsp:option selected="<%=false%>" value=
                        "<%=keyname%>"/><dsp:valueof param="key"/>
                    </core:defaultCase>
                  </core:switch>
                </dsp:getvalueof>
                </dsp:getvalueof>
                          --%>

              </dsp:oparam>
            </dsp:droplet>
            </dsp:select>
          </td>
          <td></td>
          <td>
            <%--Split the CommerceItemShippingInfos and redirect right back here
              with init=false.
            --%>

            <dsp:input bean="ShippingGroupFormHandler.splitShippingInfosSuccessURL"
              type="hidden" value="ship_to_multiple.jsp?init=false"/>
            <dsp:input bean="ShippingGroupFormHandler.ListId" paramvalue=
              "commerceItem.id" priority="<%=(int) 9%>" type="hidden"/>
            <dsp:input bean="ShippingGroupFormHandler.splitShippingInfos" type=
              "submit" value=" Save "/>
          </td>
        </tr>
        </dsp:form>
      </dsp:oparam>
    </dsp:droplet>
  </dsp:oparam>
</dsp:droplet>
```

ここでは、複合的なフォーム要素を使用します。
ShippingGroupDroplet.CommerceItemShippingInfoContainer.
CommerceItemShippingInfoMap Bean プロパティは、CommerceItemShippingInfo Lists のマップです。

要求パラメータ commerceItem.id の値は、外側の ForEach ループ内の現在の CommerceItem に対応する CommerceItemShippingInfo List を取得するための動的キーです。特定の CommerceItemShippingInfo リストへの参照を取得するために、リストのキーを使用して ShippingGroupFormHandler.listId プロパティを設定します。リストは、ShippingGroupFormHandler.currentList プロパティを通じて公開されます。リストを反復し、index パラメータでリスト内で参照される、各品目の splitQuantity プロパティおよび splitShippingGroupName プロパティを公開します。

ShippingGroup と CommerceItem の関連付けに問題がなければ、ユーザーは「**Continue**」ボタンをクリックして購入プロセスを続けます。これによって、ShippingGroupFormHandler.applyShippingGroups ハンドラが起動されます。ハンドラは、各 CommerceItemShippingInfo オブジェクト内の情報を収集し、適切な ShippingGroups と、各 CommerceItem との必要な関係をオーダーに追加します。ShippingGroupFormHandler の詳細は、『ATG Web Commerce Programming Guide』を参照してください。これを行う JSP を、次に示します。

```
<%--
    Invoke the applyShippingGroups handler and redirect to shipping_method.jsp
    upon success.
--%>

<dsp:form action="ship_to_multiple.jsp" method="post">
<dsp:input bean="ShippingGroupFormHandler.applyShippingGroupsSuccessURL"
   type="hidden" value="shipping_method.jsp"/>
<dsp:input bean="ShippingGroupFormHandler.applyShippingGroups" type="submit"
   value="Continue"/>
</dsp:form>
```

ユーザーは checkout/shipping_method.jsp にリダイレクトされ、そこで各 ShippingGroup の出荷方法を選択します。

使用可能な出荷方法

Motorprise では atg/commerce/pricing/AvailableShippingMethods サーブレット Bean を使用して、ユーザーに使用可能な出荷方法のリストを提供します。

AvailableShippingMethods サーブレット Bean が ShippingPricingEngine に問合せを行い、特定の ShippingGroup で使用可能なすべての出荷方法を決定します。ShippingPricingEngine は登録されている出荷カルキュレータを反復し、返されたそれぞれの出荷方法を含めます。Motorprise は、Ground、TwoDay および NextDay という出荷方法に対応する、3 種類の登録済 ShippingPricingCalculators を使用して出荷を行います。ユーザーの操作が終了すると、フォームのアクションによってユーザーは購入プロセスの次の部分にリダイレクトされ、そこで支払情報を指定します。

支払情報

Motorprise ユーザーは、1 つ以上の支払方法を使用して各オーダーの支払を行うことができます。各支払方法は請求先所在地に関連付けられています。支払方法は、オーダー全体に適用することも、明細品目、送料、税金など、オーダーの各部に適用することもできます。Motorprise では次の支払方法が有効です。

- クレジット・カード（調達カードまたは法人購買カード）
- 請求書要求

ユーザーには、使用が許可されている支払方法のみが表示されます。ユーザーに請求書要求が許可されている場合は、必要なだけ PO 番号または購買依頼番号を入力できます。クレジット・カードの使用が許可されている場合は、許可されているクレジット・カードのリストからそのクレジット・カードを使用できます。Motorprise では、ユーザーが新規のクレジット・カードを入力することはできません。会社の管理者に連絡して、新規支払方法を要求する必要があります。ビジネス・モデルに応じて、支払方法に異なる制限を設定できます。

Checkout/billing.jsp は、最初に、AuthorizedPaymentTypesDroplet サーブレット Bean を使用して、ユーザーが使用できる PaymentGroup のタイプを決定します。AuthorizedPaymentTypesDroplet は、クレジット・カードと請求書のどちらかの使用が許可されているユーザー、両方の使用が許可されているユーザー、またはどちらの使用も許可されていないユーザーを識別します。続いて、サーブレット Bean は、PaymentGroup タイプに適した支払フォームが含まれる特定のページ・フラグメントを埋め込みます。

```
<dsp:droplet name="AuthorizedPaymentTypesDroplet">
      <dsp:param bean="Profile" name="profile"/>
      <dsp:oparam name="output">
        <dsp:droplet name="Switch">
          <dsp:param name="value" param="potentialPaymentTypes.creditCard"/>
          <dsp:oparam name="true">
            <dsp:droplet name="Switch">
              <dsp:param name="value"
                param="potentialPaymentTypes.invoiceRequest"/>
              <dsp:oparam name="true">
                <dsp:droplet name="IsEmpty">
                  <dsp:param bean="Profile.paymentTypes" name="value"/>
                  <dsp:oparam name="true">
                    <dsp:include page="billing_invoice.jsp" flush="true">
                      <dsp:param name="init" value="true"/></dsp:include>
                  </dsp:oparam>
                  <dsp:oparam name="false">
                    <dsp:include page="billing_invoice_cc.jsp" flush="true">
                      <dsp:param name="init" value="true"/></dsp:include>
                  </dsp:oparam>
                </dsp:droplet>
              </dsp:oparam>
              <dsp:oparam name="false">
                <dsp:droplet name="IsEmpty">
                  <dsp:param bean="Profile.paymentTypes" name="value"/>
                  <dsp:oparam name="true">
                    <dsp:include page="no_billing.jsp" flush="true"></dsp:include>
                  </dsp:oparam>
```

```
                <dsp:oparam name="false">
                  <dsp:include page="billing_cc.jsp" flush="true"><dsp:param
                    name="init" value="true"/></dsp:include>
                </dsp:oparam>
              </dsp:droplet>
            </dsp:oparam>
          </dsp:droplet>
        </dsp:oparam>
        <dsp:oparam name="false">
          <dsp:droplet name="Switch">
            <dsp:param name="value" param=
              "potentialPaymentTypes.invoiceRequest"/>
            <dsp:oparam name="true">
              <dsp:include page="billing_invoice.jsp" flush="true"><dsp:param
                name="init" value="true"/></dsp:include>
            </dsp:oparam>
            <dsp:oparam name="false">
              <dsp:include page="no_billing.jsp" flush="true"></dsp:include>
            </dsp:oparam>
          </dsp:droplet>
        </dsp:oparam>
      </dsp:droplet>
    </dsp:oparam>
  </dsp:droplet>
```

購入プロセスの請求部分では、PaymentGroup オブジェクトを使用して支払情報をカプセル化しています。各 PaymentGroup とオーダーの間の関連付けは、CommerceIdentifierPaymentInfo オブジェクトで保持されます。様々な請求ページ・フラグメントで、PaymentGroupDroplet を使用してユーザーの PaymentGroups および CommerceIdentifierPaymentInfos を作成します。これらは、購入プロセス全体をとおしてこれらのオブジェクトを保持するセッション・コンポーネントである、ユーザーの PaymentGroupMapContainer および CommerceIdentifierPaymentInfoContainer に追加されます。

クレジット・カード

クレジット・カードの使用がユーザーに許可されている場合、checkout/billing_cc.jsp で PaymentGroupDroplet が使用されます。コードのスニペットを、次に示します。

```
<%--
    The PaymentGroupDroplet is used here to initialize the user's CreditCard
    PaymentGroups, and an OrderPaymentInfo object to associate a PaymentGroup with
    the Order.
--%>

<dsp:droplet name="PaymentGroupDroplet">
  <dsp:param name="clear" param="init"/>
  <dsp:param name="paymentGroupTypes" value="creditCard"/>
  <dsp:param name="initPaymentGroups" param="init"/>
  <dsp:param name="initOrderPayment" param="init"/>
  <dsp:oparam name="output">
```

PaymentGroupDroplet 初期化はコンテキストに依存します。そのため、このページは、以前の PaymentGroupMapContainer および CommerceIdentifierPaymentInfoContainer を消去し、再初期化する必要があるかどうかを指定する、init という初期化パラメータを受け取ります。初期化の際、PaymentGroupDroplet はユーザーの paymentTypes プロファイル・プロパティに対応する CreditCard PaymentGroups を作成し、オーダー全体の支払情報を表す OrderPaymentInfo を作成します。ユーザーが PaymentGroup を選択し、「**Continue**」をクリックすると、PaymentGroupFormHandler.applyPaymentGroups ハンドラが起動されます。これによって moveToConfirmation パイプライン・チェーンが実行され、ユーザーは購入プロセスの次の部分にリダイレクトされます。

請求書

クレジット・カードではなく請求書の使用がユーザーに許可されている場合、PaymentGroupDroplet は、作成する PaymentGroup タイプを指定する異なるパラメータを使用して初期化されます。PaymentGroupDroplet は、checkout/billing_invoice.jsp で次のように記述されています。

```
<%--
    The PaymentGroupDroplet is used here to initialize an InvoiceRequest
    PaymentGroup, and an OrderPaymentInfo object to associate a PaymentGroup with
    the Order.
--%>
<dsp:droplet name="PaymentGroupDroplet">
  <dsp:param name="clear" param="init"/>
  <dsp:param name="paymentGroupTypes" value="invoiceRequest"/>
  <dsp:param name="initPaymentGroups" param="init"/>
  <dsp:param name="initOrderPayment" param="init"/>
  <dsp:oparam name="output">
```

請求書およびクレジット・カード

checkout/billing_invoice_cc.jsp に示されているように、請求書とクレジット・カードの両方の使用がユーザーに許可される場合があります。

```
<%--
    The PaymentGroupDroplet is used here to initialize the user's CreditCard
    PaymentGroups and an InvoiceRequest PaymentGroup, as well as an
    OrderPaymentInfo object to associate a PaymentGroup with the Order.
--%>

<dsp:droplet name="PaymentGroupDroplet">
  <dsp:param name="clear" param="init"/>
  <dsp:param name="paymentGroupTypes" value="creditCard,invoiceRequest"/>
  <dsp:param name="initPaymentGroups" param="init"/>
  <dsp:param name="initOrderPayment" param="init"/>
  <dsp:oparam name="output">
```

オーダーの価格の再設定

精算プロセスにおいて、ユーザーはカートや出荷情報などを変更することができ、その結果、オーダーの価格に影響を及ぼすことがあります。Oracle Commerce Core Commerce には、オーダーの価格を再設定するための様々な方法があります。たとえば、CartModifierFormHandler クラスは、ショッピング・カートが変更されるとオーダーの価格を自動的に再設定します。しかし、購入プロセスの他のフォーム・ハンドラはオーダーの価格を自動的に再設定しないため、明示的に価格の再設定を行う必要があります。

Motorprise の精算プロセスでは、ユーザーは請求情報の前に出荷情報を発行します。所在地や出荷方法などの情報を変更できるため、これが送料、さらにオーダー合計価格に影響する可能性があります。出荷ページで使用される ShippingGroupFormHandler が、オーダーの価格を自動的に再設定することはありません。そのため、請求ページでオーダーの価格を明示的に再設定して、新規出荷情報を価格に反映させる必要があります。

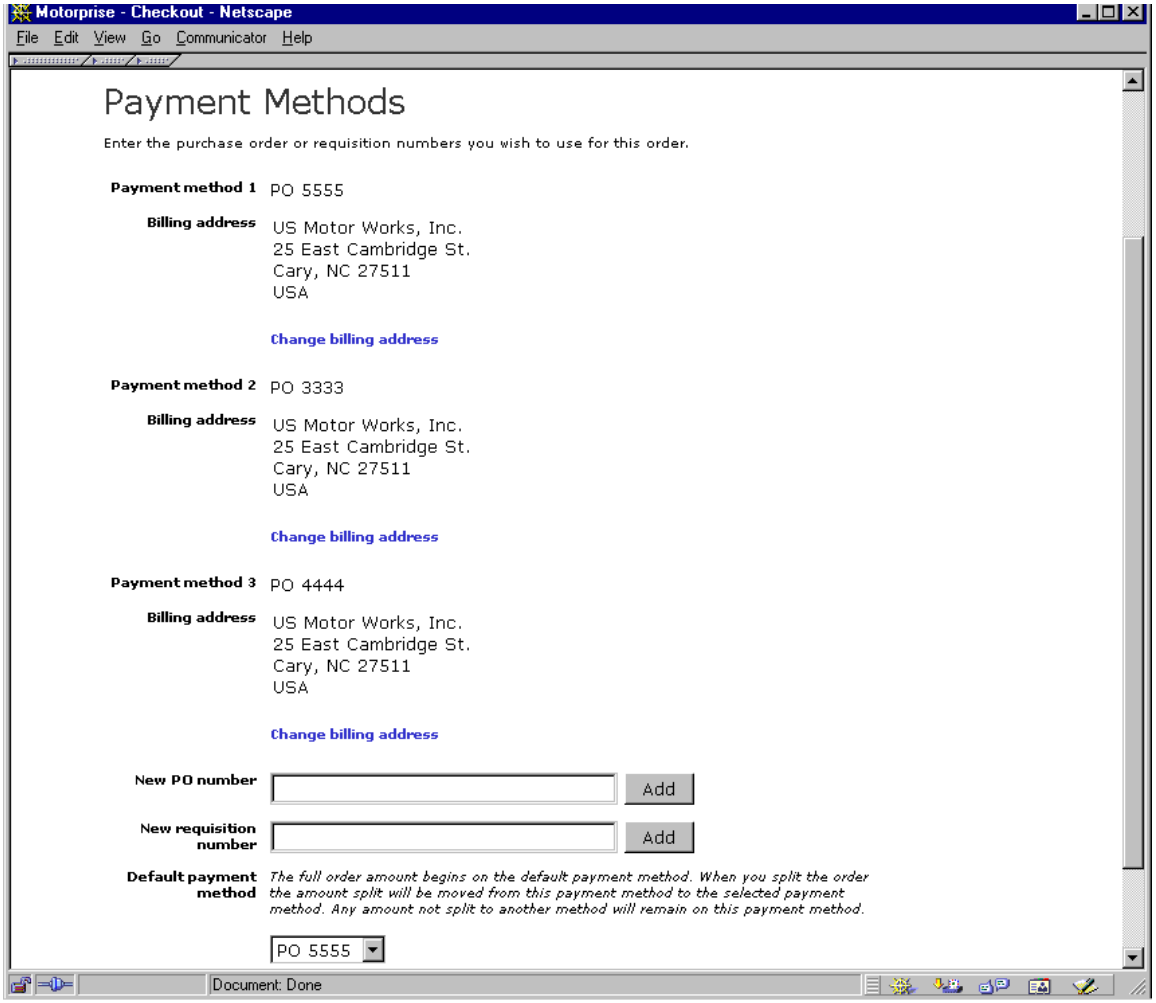
オーダーの価格を再設定するために、billing_invoice.jsp、billing_invoice_cc.jsp および billing_cc.jsp といった請求ページの先頭で RepriceOrderDroplet が次のように使用されています。

```
<dsp:importbean bean="/atg/commerce/order/purchase/RepriceOrderDroplet"/>

<%--Reprice the Order total so that we can assign PaymentGroups to any
CommerceIdentifier.
--%>
<dsp:droplet name="RepriceOrderDroplet">
  <dsp:param name="pricingOp" value="ORDER_TOTAL"/>
</dsp:droplet>
```

複数の支払グループの選択

Motorprise では、複数の支払方法を使用してオーダーを処理するオプションがユーザーに提供されます。この機能は、billing.jsp からリンクされる、checkout/payment_methods.jsp をはじめ、Motorprise に実装されています。このページは、オーダーの特定の部分に支払方法が実際に適用される前に、ユーザーが使用できる支払方法のリストを収集するために使用されます。

この機能もやはり、ユーザーに使用が許可されている PaymentGroups のタイプに依存します。請求書の使用が許可されている場合、ユーザーは複数の PO 番号または購買依頼番号を入力できます。クレジット・カードの使用が許可されている場合は、プロファイル内にあるすべてのカードを使用できます。ユーザーは、分割支払ページに進む前に、まずデフォルト PaymentGroup を選択します。すべてのオーダー・コストは、最初、このデフォルト PaymentGroup に適用されます。次に、そのコストの各部を異なる PaymentGroups に分割できます。オーダー全体に適用される PaymentGroups をユーザーが分割できるように、checkout/SplitPaymentOrderDetails.jsp が使用されています。



複数の支払方法の追加およびデフォルト支払方法の選択

PaymentGroupDroplet を使用して必要なヘルパー・オブジェクトを提供し、ユーザーの許可に基づいて、ユーザーのクレジット・カードを PaymentGroupMapContainer に含めます。以前の payment_methods.jsp で作成された InvoiceRequest オブジェクトは、session-scoped のコンテナ内にすでにあります。次の SplitPaymentOrderDetails.jsp からの JSP が、この初期化を行います。

```
<%--
    The PaymentGroupDroplet initializes an OrderPaymentInfo object based on
    the value of the request parameter "init". CreditCard PaymentGroups are also
    initialized if the user is authorized to use them.
--%>
<dsp:droplet name="PaymentGroupDroplet">
   <dsp:param name="initOrderPayment" param="init"/>
   <dsp:param name="paymentGroupTypes" value="creditCard"/>
   <dsp:param name="initPaymentGroups" param="init"
   <dsp:oparam name="output">
```

初期化が完了すると、オーダーの支払情報を編集するフォームがユーザーに表示されます。OrderPaymentInfo ヘルパー・オブジェクトには、PaymentGroupMapContainer 内の PaymentGroups を、金額で Order に関連付ける情報が格納されます。

似たページが checkout/SplitPaymentDetails.jsp です。これは SplitPaymentOrderDetails.jsp と同じように動作しますが、ユーザーが明細品目、送料および税金で PaymentGroups を分割できる点が異なります。PaymentGroupDroplet によって、オーダー内の各 CommerceItem の CommerceItemPaymentInfo オブジェクト、Order 内の各 ShippingGroup の ShippingGroupPaymentInfo オブジェクト、および税金の TaxPaymentInfo が作成されます。次の JSP がこの初期化を行います。

```
<%--The PaymentGroupDroplet initializes the CommerceIdentifierPaymentInfo objects
    based on the value of the request parameter "init". CreditCard PaymentGroups
    are also initialized if the user is authorized to use them.
--%>

<dsp:droplet name="PaymentGroupDroplet">
  <dsp:param name="clear" param="init" />
  <dsp:param name="paymentGroupTypes" value="creditCard,storeCredit" />
  <dsp:param name="initPaymentGroups" param="init" />
  <dsp:param name="initBasedOnOrder" param="init" />
  <dsp:param name="initOrderPayment" param="init" />
  <dsp:param name="createAllPaymentInfos" value="true" />
  <dsp:oparam name="output">
  </dsp:oparam>
</dsp:droplet>
```

オーダーの OrderPaymentInfo オブジェクトを反復して、フォームを作成します。最初、PaymentGroupDroplet は、全体の Order コストをデフォルト PaymentGroup に関連付ける 1 つの OrderPaymentInfo オブジェクトを作成します。

ユーザーが異なる支払方法を使用してオーダーの各部を分割するたびに、新規 OrderPaymentInfo オブジェクトが CommerceIdentifierPaymentInfoContainer に追加されます。

オーダーを金額によって複数の支払方法に分割する。

ユーザーはこのフォームで、これらのオブジェクトの `SplitAmount` および `SplitPaymentMethod` プロパティ値を更新し、`PaymentGroupFormHandler.splitPaymentInfo` をコールすることによってこれらを発行できます。このプロパティは、オーダー・コストを、`PaymentGroupDroplet` の初期化で提供されるものではなく、追加または異なる `PaymentGroups` に関連付けます。

支払タイプおよび支払額を含む明細品目として表示される、これらのすべての `CommerceIdentifierPaymentInfo` オブジェクトを編集するフォームが、ユーザーに表示されます。最初に、`CommerceItems`、`ShippingGroups` および `Order` 自体を取得します。それぞれについて、`CommerceItemPaymentInfo` オブジェクトの対応するリストを取得します。各リストを反復し、入力を提供して、品目の `SplitAmount` および `SplitPaymentMethod` プロパティ値を編集します。

複合的な `ShippingGroups` の場合と同様、ここでも複合的なフォーム要素のセットが発生します。`PaymentGroupDroplet.CommerceIdentifierPaymentInfoContainer.CommerceIdentifierPaymentInfoMap` Bean プロパティは、`CommerceIdentifierPaymentInfo Lists` の `Map` です。要求パラメータ `commerceItem.id` の値は、外側の `ForEach` ループ内の現在の `CommerceItem` に対応する `CommerceIdentifierPaymentInfo List` を取得するための動的キーです。特定の `CommerceIdentifierPaymentInfo` リストへの参照を取得するために、リストのキーを使用して `PaymentGroupFormHandler.listId` プロパティを設定します。リストは、`PaymentGroupFormHandler.currentList` プロパティを通じて公開されます。リストを反復し、`index` パラメータでリスト内で参照される、各品目の `splitAmount` プロパティおよび `splitPaymentMethod` プロパティを公開します。

支払方法の関連付けに問題がなければ、ユーザーは「**Continue**」をクリックして購入プロセスを続けます。`PaymentGroupFormHandler.applyPaymentGroups` ハンドラが起動されます。ハンドラは、`CommerceIdentifierPaymentInfo` オブジェクト内の情報を収集し、適切な `PaymentGroups` と、オーダー内の `CommerceItem` との必要な関係を `Order` に追加します。`PaymentGroupFormHandler` の詳細は、『ATG Web Commerce Programming Guide』を参照してください。これを行う JSP を、次に示します。

```
<dsp:form action="split_payment.jsp" method="post">
<dsp:input bean="PaymentGroupFormHandler.applyPaymentGroupsSuccessURL" type=
   "hidden" value="IsEmptyCostCenters.jsp?link=split_payment.jsp"/>
<dsp:input bean="PaymentGroupFormHandler.applyPaymentGroups" type="submit"
   value="Continue"/>
</dsp:form>
```

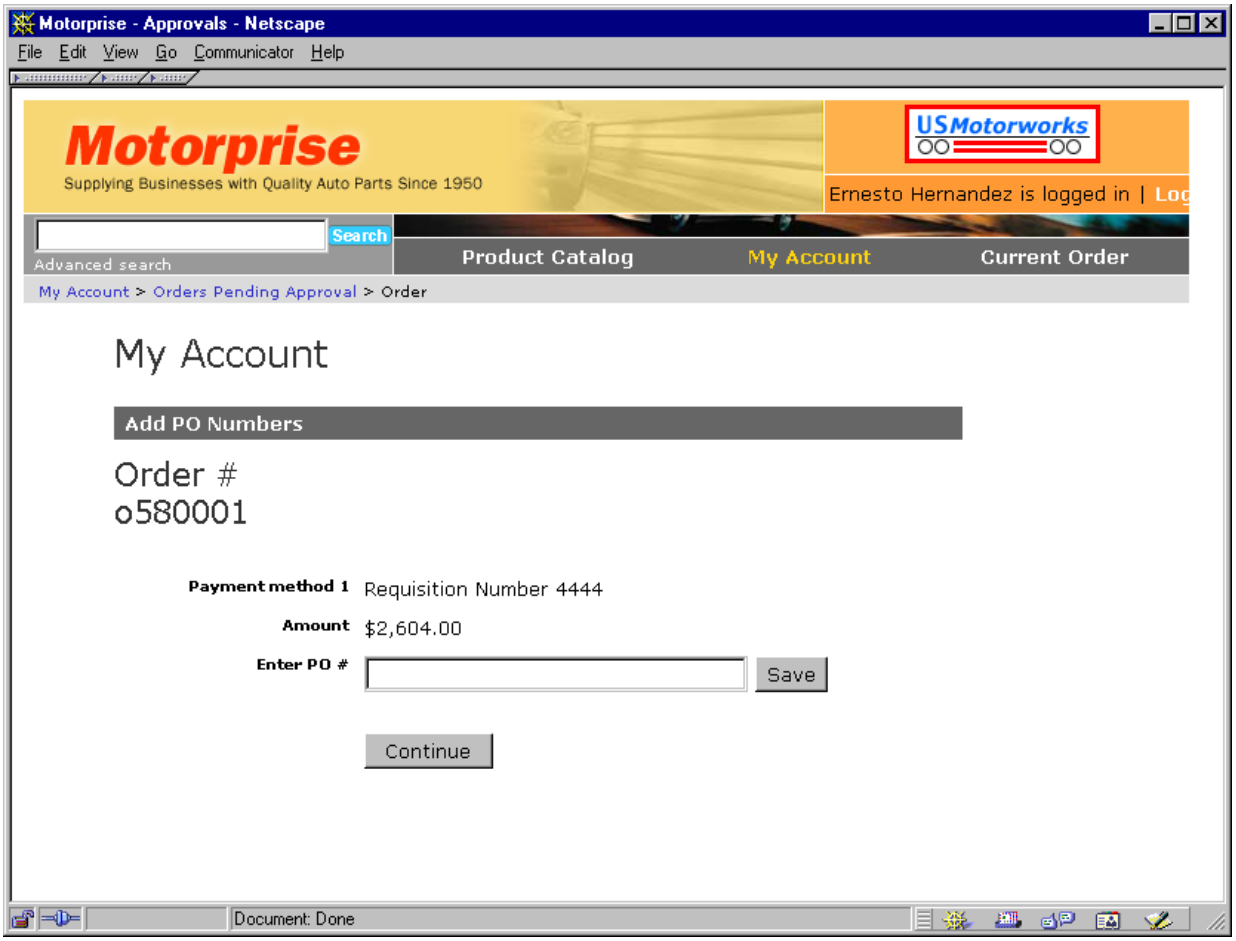
購買依頼

Motorprise では、ユーザーが精算し、請求書を要求するときに、PO 番号 *または*購買依頼番号を指定できます。後者を選択した場合は、後でオーダーが承認されるときに PO 番号を指定します。Motorprise では、ユーザーのプロファイル内の `requireApproval` プロパティが `false` の場合を除き、購買依頼番号を使用するオーダーには自動的に承認が必要です。ビジネス・モデルに応じて、異なる承認条件を設定できます。

B2B コマース・プラットフォームでは、バイヤーは請求書による支払の場合は常に、購買オーダー番号を指定する必要があります（購買依頼番号を使用している場合でも）。Motorprise ストアではこの動作が上書きされており、バイヤーは購買オーダー番号、購買依頼番号またはその両方を指定できます。オーダーの作成時にバイヤーが購買オーダー番号を指定しない場合は、承認者がオーダーの承認時にそれを指定する必要があります。

承認パイプライン・マネージャ内の `checkRequiresApproval` チェーンには、別のパイプライン・プロセッサが追加されています。これが、オーダー内のすべての支払グループを反復して、購買依頼番号を検索します。これが見つかると、パイプライン結果オブジェクトにエラーが追加されて、承認が必要であることを承認システムに対して指示します。

購買依頼番号はあるが購買オーダー番号がないオーダーを承認者が受け取ると、`user/add_po_number.jsp` で P.O.番号を入力することが要求されます。



購買オーダー番号がないオーダーは承認者がその番号を入力する必要がある。

コスト・センターの指定

オーダー内の品目に対して社内で請求が行われる特定のコスト・センターを指定することもできます。コスト・センターはビジネス・ユニット・レベルで定義され、ユーザーによって継承されます。

精算プロセスの支払手順の後、ユーザーはコスト・センターとオーダーの関連付けについてたずねられることがあります。この決定は `checkout/IsEmptyCostCenters.jsp` で行われます。ユーザーがコスト・センターを持たない場合は、`checkout/confirmation.jsp` にリダイレクトされます。コスト・センターを持つ場合は、`Redirect` ドロップレットが `checkout/cost_centers.jsp` に進みます。

ユーザーがコスト・センターを持つ場合、それをオーダー全体またはオーダーの各部に割り当てることができます。ユーザーが選択できるコスト・センターのリストには、デフォルト・コスト・センター、および許可されているコスト・センターの代替リストが含まれています。単純な例では、ユーザーはオーダー内のすべての品目を、(`checkout/cost_centers.jsp` 内でデフォルトで選択される) デフォルト・コスト・センターに割り当てることや、選択ボックスから別のコスト・センターを選択することができます。

`checkout/cost_centers.jsp` は、`CostCenterDroplet` を使用して、特定のユーザーに適したコスト・センターおよび `CommerceIdentifierCostCenters` を作成します。これは `checkout/cost_centers.jsp`

の中で次のように使用されて、許可されているすべてのコスト・センターを選択ボックスに表示し、コスト・センターをオーダーに割り当てます。

```
<%--
    CostCenterDroplet initializes cost centers available to user, and also
    CostCenters objects corresponding to the order to associate cost centers to
    the order
--%>

<dsp:droplet name="CostCenterDroplet">
<dsp:param name="clear" value="true"/>
<dsp:param name="initCostCenters" value="true"/>
<dsp:param name="initItemCostCenters" value="false"/>
<dsp:param name="initShippingCostCenters" value="false"/>
<dsp:param name="initTaxCostCenters" value="false"/>
<dsp:param name="initOrderCostCenters" value="true"/>
<dsp:param name="useAmount" value="false"/>
<dsp:oparam name="output">

  <tr valign=top>
    <td></td>
        <td align=right><span class=smallb>Cost center</span></td>
    <td>
      <dsp:setvalue bean="CostCenterFormHandler.listId" paramvalue="order.id"/>
      <dsp:input bean="CostCenterFormHandler.listId" paramvalue="order.id"
        priority="<%=(int) 9%>" type="hidden"/>
        <%-- we only expect this to have 1 element at [0], but we put this in a
          ForEach to be safe --%>
      <dsp:droplet name="ForEach">
        <dsp:param bean="CostCenterFormHandler.currentList" name="array"/>
        <dsp:oparam name="output">
          <dsp:select bean=
            "CostCenterFormHandler.currentList[param:index].CostCenterName"><br>

            <%--
                 List all the cost centers available to the user in select
                 box, so he can choose one among them.
            --%>
          <dsp:droplet name="ForEach">
            <dsp:param bean="Profile.costCenters" name="array"/>
            <dsp:param name="elementName" value="CostCenter"/>
            <dsp:oparam name="output">
              <dsp:droplet name="Switch">
                <dsp:param name="value" param="CostCenter.repositoryId"/>
                <dsp:getvalueof id="costCenterId" idtype="String"
                  param="CostCenter.identifier">
     <dsp:getvalueof id="defaultCostCenter" idtype="String"
       bean="Profile.defaultCostCenter.repositoryId">
                 <dsp:oparam name="<%=defaultCostCenter%>">
```

```
                    <dsp:option selected="<%=true%>" value="<%=costCenterId%>"/>
                      <dsp:valueof param="CostCenter.identifier"/> -
                      <dsp:valueof param="CostCenter.description"/>
                  </dsp:oparam>
                  <dsp:oparam name="default">
                    <dsp:option selected="<%=false%>" value="<%=costCenterId%>"/>
                      <dsp:valueof param="CostCenter.identifier"/>
                      <dsp:valueof param="CostCenter.description"/>
                  </dsp:oparam>
                </dsp:getvalueof>
       </dsp:getvalueof>

          </dsp:droplet> <%-- Switch --%>
    </dsp:oparam><%-- End: ForEach.oparam --%>
    </dsp:droplet> <%-- End: ForEach --%>
          </dsp:select>
        </dsp:oparam>
      </dsp:droplet>
  </td></tr>
    </dsp:oparam>
    </dsp:droplet> <%-- End: CostCenterDroplet --%>
```

ユーザーは任意のコスト・センターをオーダーに割り当てることができ、フォームを発行することによって、購入プロセスの次の手順（確認）に進むことができます。

複数のコスト・センターの使用

オーダー内の品目間でのコスト・センターの分割は、cost_centers.jsp からリンクされている cost_centers_line_item.jsp で実装されます。

このページでは、要求パラメータ init を使用して、CostCenterDroplet の初期化を有効または無効にします。オーダー内のすべての品目、出荷グループおよび税金に対してコスト・センターを作成するには、costCentersLineItemDetails.jsp で CostCenterDroplet を使用します。PaymentGroupDroplet と同様、CostCenterDroplet はコンテキストに依存します。このドロップレットは、以前の CostCenterMapContainer および CommerceIdentiferCostCenterMap を再初期化する必要があるかどうかを指定する、init パラメータを受け取ります。CostCenterDroplet は、costCentersLineItemDetails.jsp で次のように記述されています。

```
<%--
     CostCenterDroplet initializes the CommerceIdentifierCostCenter objects based
     on the requested init parameter and also creates costcenter objects.
--%>

<dsp:droplet name="CostCenterDroplet">
  <dsp:param name="clear" param="init"/>
  <dsp:param name="initCostCenters" value="true"/>
  <dsp:param name="initItemCostCenters" param="init"/>
  <dsp:param name="initShippingCostCenters" param="init"/>
  <dsp:param name="initTaxCostCenters" param="init"/>
```

```
  <dsp:param name="useAmount" value="false"/>
  <dsp:oparam name="output">
```

初期化が完了すると、コスト・センターを編集するフォームがユーザーに表示されます(これによって、CommerceIdentifierCostCenter オブジェクトが変更されます)。最初に、オーダーから CommerceItems、ShippingGroups および税金グループを収集します。それぞれについて、CommerceIdentiferCostCenter オブジェクトの対応するリストを取得し、各リストを反復します。

ドロップダウン・リストが提供され、ユーザーはオーダー内の商品に対応する CommerceIdentiferCostCenter オブジェクトの splitCostCenterName プロパティと splitQuantity プロパティ、および出荷グループおよび税金用の品目の splitCostCenterName を編集できます。

次のコードは、オーダー内の CommerceItems の CommerceIdentifierCostCenter オブジェクトを変更するフォームを示しています。

```
<%--
    Iterate through the commerceitems of the order and assign that particular
    commerceIdentifierCostCenter to the CostCenterFormHandler to edit the
    costcenter information. Since we provide <dsp:form></dsp:form> elements across
    each item we will be editing only one item at any given time.
    If user splits any item across cost centers, we create extra
    CommerceIdentifierCostCenter to accomodate the split qty.
--%>

<dsp:droplet name="ForEach">
  <dsp:param name="array" param="order.commerceItems"/>
  <dsp:oparam name="output">
    <%--
      Set the current item id to CostCenterFormHandler
      --%>
    <dsp:setvalue paramvalue="element" param="commerceItem"/>
    <dsp:setvalue bean="CostCenterFormHandler.listId"
      paramvalue="commerceItem.id"/>
        <dsp:droplet name="ForEach">
          <dsp:param bean="CostCenterFormHandler.currentList" name="array"/>
          <dsp:oparam name="output">
            <%-- begin line item --%>

            <tr valign=top>
              <dsp:form action="cost_centers_line_item.jsp" method="post">
              <td><dsp:valueof param="commerceItem.catalogRefId"/></td>
              <td></td>
              <td><dsp:a href="../catalog/product.jsp?navAction=jump">
                     <dsp:param name="id" param=
                       "commerceItem.auxiliaryData.productId"/>
                     <dsp:valueof param=
                       "commerceItem.auxiliaryData.productRef.displayName"/>
                       </dsp:a></td>
              <td></td>
```

```
<td align=middle><dsp:valueof param="element.quantity"/></td>
<td> </td>


<td>
<dsp:input bean=
  "CostCenterFormHandler.currentList[param:index].splitQuantity"
  paramvalue="element.quantity" size="4" type="text"/></td>
<td> </td>


<td align=center>
    <%--
      Set the cost center to be used for the current item
      --%>
  <dsp:getvalueof id="itemCenterName" idtype="String" param=
    "element.costCenterName">
  <dsp:select bean="CostCenterFormHandler.
    currentList[param:index].splitCostCenterName">
    <%--
      Iterate through the available cost centers so that user can
      choose one among them to assign for the current item
      --%>
  <dsp:droplet name="ForEach">
    <dsp:param name="array" param="costCenters"/>
    <dsp:oparam name="output">
      <dsp:droplet name="Switch">
        <dsp:param name="value" param="key"/>
        <dsp:getvalueof id="keyname" idtype="String" param="key">
       <%-- <dsp:oparam name="param:...element.costCenterName"> --%>
          <dsp:oparam name="<%=itemCenterName%>">
          <dsp:option selected="<%=true%>" value="<%=keyname%>"/>
            <dsp:valueof param="key"/>
          <dsp:droplet name="ForEach">
           <dsp:param bean="Profile.costCenters" name="array"/>
           <dsp:param name="elementName" value="costCenter"/>
           <dsp:oparam name="output">
            <dsp:droplet name="Switch">
             <dsp:param name="value" param="costCenter.identifier"/>
             <dsp:oparam name="<%=keyname%>">
              <dsp:valueof param="costCenter.description"/>
             </dsp:oparam>
            </dsp:droplet>
           </dsp:oparam>
          </dsp:droplet>
       </dsp:oparam>
       <dsp:oparam name="default">
         <dsp:option selected="<%=false%>" value="<%=keyname%>"/>
```

```
                        <dsp:valueof param="key"/>
                       <dsp:droplet name="ForEach">
                        <dsp:param bean="Profile.costCenters" name="array"/>
                        <dsp:param name="elementName" value="costCenter"/>
                        <dsp:oparam name="output">
                         <dsp:droplet name="Switch">
                          <dsp:param name="value" param="costCenter.identifier"/>
                          <dsp:oparam name="<%=keyname%>">
                           <dsp:valueof param="costCenter.description"/>
                          </dsp:oparam>
                         </dsp:droplet>
                        </dsp:oparam>
                       </dsp:droplet>
                      </dsp:oparam>
                      </dsp:getvalueof>
                    </dsp:droplet>
                  </dsp:oparam>
                </dsp:droplet>
                </dsp:select>

                </dsp:getvalueof>
              </td>
              <td align=center>
                <dsp:input bean="CostCenterFormHandler.listId" paramvalue=
                  "commerceItem.id" priority="<%=(int) 9%>" type="hidden"/>
                <dsp:input bean="CostCenterFormHandler.splitCostCentersSuccessURL"
                  type="hidden" value="cost_centers_line_item.jsp?init=false"/>
                <dsp:input bean="CostCenterFormHandler.splitCostCentersErrorURL"
                  type="hidden" value="cost_centers_line_item.jsp?init=false"/>
                <dsp:input bean="CostCenterFormHandler.splitCostCenters"
                  type="submit" value="Save"/>
              </td>
            </tr>
            </dsp:form>
            <%-- end line item --%>
          </dsp:oparam>
        </dsp:droplet>
  </dsp:oparam>
</dsp:droplet>
```

フォームを発行すると、フォームのアクションによってユーザーは、checkout/confirmation.jsp で始まる購入プロセスの次の部分にリダイレクトされます。

オーダーの確認および確定

オーダー確認ページ checkout/confirmation.jsp には、出荷および支払情報、最終的な合計額など、オーダー全体が表示されます。ユーザーは次の操作を行うことができます。

- その時点でのオーダーの発行
- オーダーの取消し
- 将来のある時点でのオーダーの発行のスケジュール

ユーザーが確認画面でオーダーを発行すると、オーダーの承認条件の有無がチェックされます。オーダーに承認が必要な場合は、オーダーは承認保留状態に置かれます。承認の詳細は、「自分のアカウント」を参照してください。

オーダーに承認が必要ない場合は、発行されて処理が行われます。

確認ページ checkout/confirmation.jsp では、CommitOrderFormHandler および CancelOrderFormHandler を使用して、オーダーの発行または取消しをそれぞれ行います。

```
<dsp:input bean="CommitOrderFormHandler.commitOrder" type="submit" value="Place
  order now"/>
<dsp:input bean="CommitOrderFormHandler.commitOrderSuccessURL" type="hidden"
  value="../checkout/thank_you.jsp"/>
<dsp:input bean="CommitOrderFormHandler.commitOrderErrorURL" type="hidden"
  value="../checkout/confirmation.jsp"/>
<dsp:input bean="CancelOrderFormHandler.cancelOrder" type="submit" value="Cancel
  order"/> <dsp:input bean="CancelOrderFormHandler.orderIdToCancel" beanvalue=
  "ShoppingCart.current.id" type="hidden"/>
```

定期オーダーの詳細は、「自分のアカウント」の「定期オーダー」の項を参照してください。

オーダー確定ページにはオーダーの概要が示され、ID が割り当てられ、ユーザーへのお礼のメッセージが表示されます。

精算用のコンポーネントのバックアップ

Motorprise はデモであるため、パフォーマンス上の理由で、特定のコンポーネントがバックアップされていません。出荷および支払情報は、ユーザーがオーダーを確定するまで、一時的にメモリーに格納されます。したがって、なんらかの理由で Dynamo が停止すると、この情報は失われ、ユーザーが情報を再入力する必要があります。

簡単な構成の変更を行うことで、この情報をバックアップ・サーバーによって復元できます。/atg/dynamo/Configuration.properties に次の行を追加します。

```
sessionBackupServerPropertyList+=\
   /atg/commerce/order/purchase/ShippingGroupContainerService.shippingGroupMap,\
   /atg/commerce/order/purchase/ShippingGroupContainerService.
       commerceItemShippingInfoMap,\
   /atg/commerce/order/purchase/ShippingGroupContainerService.
       defaultShippingGroupName,\
   /atg/commerce/order/purchase/PaymentGroupContainerService.paymentGroupMap,\
   /atg/commerce/order/purchase/PaymentGroupContainerService.
       commerceIdentifierPaymentInfoMap,\
   /atg/commerce/order/purchase/PaymentGroupContainerService.
       defaultPaymentGroupName
```

10 SOAP サポート

多くの B2B Web サイトと同様、Motorprise は、通販代行会社からパートナの業務システムにいたるまで、他の多くのシステムと統合する必要があります。Oracle Commerce ビジネス・コマース・リファレンス・アプリケーションの Motorprise の例は、この統合を実現できる 1 つの方法を示しています。Motorprise には、Simple Object Access Protocol（SOAP）を介して XML ドキュメントを交換する簡単な例が含まれています。通信レイヤーのプロトコルとして SOAP を選択したのは、SOAP が XML ベースであり、異種システムの統合をサポートするためです。この章は次の項から構成されています。

SOAP とは
Simple Object Access Protocol の概要を説明します。

Motorprise における SOAP の使用
Motorprise における SOAP の使用方法について説明します。

Motorprise における SOAP のクライアント要素
クライアント要素について説明します。

Motorprise における SOAP のサーバー要素
サーバー要素について説明します。

SOAP とは

Simple Object Access Protocol（SOAP）は、様々なシステム間の通信を実現するプロトコルです。システム間で送信できるメッセージの標準的な作成方法を記述します。SOAP は、エンベロープやボディなど、メッセージの様々な部分の標準的位置を示す、メッセージの形式を定義します。このようなメッセージ各部の場所や特性など、標準を確立することで、通信に関係するすべてのシステムが同じプロトコルを使用するため、統合を短時間で実現できます。SOAP 通信では、標準のメッセージ形式に加えて、メッセージ・ボディ自体が XML ドキュメントです。テキスト（XML）を処理できるシステムはプロトコルを理解できるため、異種システムの統合が可能です。

SOAP は、XML RPC（Remote Procedure Call）の必要性から生み出されました。すべての SOAP 相互作用には、クライアントとサーバーの 2 つの部分があります。

SOAP クライアントは次の処理を行います。

- SOAP エンベロープを生成します。
- ペイロードまたはコンテンツを配置します。
- メッセージが配信される宛先サーバー上で特定のサービスを指定します。
- HTTP を介して宛先サーバーへメッセージを送信します。

サーバーは次の処理を行います。

- サービス名をサービスの実際の場所にマップするレジストリを保持します。

- メッセージを受信し、それを調べて配信先のサービスを決定します。
- サービス名のマッピング・テーブルを調べます。
- メッセージのコンテンツをマップ・サービス内の関連メソッドに配信します。

Motorprise における SOAP の使用

多くの B2B サイトにおける統合のニーズは、システム間でのオーダー情報の通信に集中しています。たとえば、ビジネス・コマース・サイトは、通販代行会社、会計目的の金融システム、または Ariba PunchOut などの買い手側アプリケーションに、オーダーを発行する必要がある可能性があります。Motorprise ストアでは、この統合を示すためにシナリオ SendOrderViaSOAP が作成されています。

SendOrderViaSOAP は、発行されるオーダーをリスニングします。オーダーがある場合、オーダーは XML ドキュメントにシリアル化され、SOAP 要求を通じて SOAP サーバーへ送信されます。SOAP サーバーは SOAP メッセージのコンテンツを Dynamo コンソールに出力します。

Motorprise における SOAP 機能のデモ

Motorprise における SOAP の使用例を見るには、SOAP シナリオ SendOrderViaSOAP を有効にし、オーダーを発行する必要があります。

デフォルトでは、ストア内でオーダーが発行されるたびに XML がコンソールに印刷されるのを防ぐために、SendOrderViaSOAP は無効です。シナリオを有効にするには、次の操作を行います。

1. ATG Control Center で、「シナリオ」→「シナリオ」に進みます。
2. 「B2B」フォルダを選択します。
3. 「SendOrderViaSOAP」シナリオを右クリックし、「シナリオ使用可能」を選択します。

これでシナリオが有効になります。SOAP を通じて送信されるオーダーを確認するには、ストア内でオーダーを発行します（承認が不要なオーダーを発行してください。たとえば、Stuart が発行するオーダーは承認が必要ありません）。シナリオはオーダーの発行をリスニングし、それを自動的に XML にシリアル化し、送信します。コンソールに印刷される SOAP メッセージを確認できます。

Motorprise における SOAP のクライアント要素

Motorprise では、一連のパラメータを SOAP メッセージとして送信する単純な SOAP クライアントが作成および構成されています。作成されたクラスは、`atg.projects.b2bstore.soap.SimpleSOAPClient` にあります。このクラスを使用すると、SOAP クライアントおよびその共通して変更される値を簡単に構成できます。

`<ATG11dir>/MotorpriseJSP/j2ee-apps/motorprise/config/atg/projects/b2bstore/soap/SimpleSOAPClient.properties`:

```
$class=atg.projects.b2bstore.soap.SimpleSOAPClient

SOAPServerURL=http\://hostname\:8080/Dynamo/solutions/B2BStore/soap
targetObjectURI=urn\:motorprise-display-xml
methodName=receiveDocument
```

プロパティ・ファイル内で、SOAP メッセージの送信先の URL は、`http://hostname:8080/Dynamo/solutions/B2BStore/soap` として定義されています。サーバーはこの URL で SOAP メッセージをリスニングします。

`targetObjectURI=urn\:motorprise-display-xml` プロパティは、メッセージのルーティング先のサービスを指定します。

最後に、`methodName=receiveDocument` プロパティは、起動する必要がある、構成されたサービス上のメソッドを指定します。この例では、これは、XML ドキュメントを Dynamo コンソールに表示する `receiveDocument` メソッドです。

SendOrdersViaSOAP シナリオには、次のカスタム・アクションが関連付けられています: `atg.projects.b2bstore.soap.SendObjectAsXML`。これは、オーダー・オブジェクトを XML ドキュメントとして送信する、シナリオの一部です。

このカスタム・アクションを Motorprise で実装するために、シナリオ・サーバーを制御する構成ファイル `/atg/scenario/scenarioManager.xml` が変更されています。ファイルに次の行が追加されています。

```
<!-- Action to send an object over a SOAP request -->
<!-- Used in Motorprise to demonstrate sending of -->
<!-- orders as an XML document.                   -->
<action>
  <action-name>
    Send Object As XML via SOAP
  </action-name>
  <action-class>
    atg.projects.b2bstore.soap.SendObjectAsXML
  </action-class>
  <resource-bundle>
    atg.projects.b2bstore.scenario.UserResources
  </resource-bundle>
  <display-name-resource>
    sendObjectAsXML.displayName
  </display-name-resource>
  <description-resource>
    sendObjectAsXML.description
  </description-resource>
  <action-execution-policy>
    collective
  </action-execution-policy>
  <action-error-response>
    delete
  </action-error-response>

  <!-- Parameter that indicates the object that will -->
  <!-- be marshalled to XML.                         -->
  <action-parameter>
    <action-parameter-name>
      marshalObject
    </action-parameter-name>
    <display-name-resource>
```

```
      sendObjectAsXML.marshalObject.displayName
    </display-name-resource>
    <action-parameter-class>
      java.lang.Object
    </action-parameter-class>
    <required>
      true
    </required>
    <description-resource>
      sendObjectAsXML.marshalObject.description
    </description-resource>
  </action-parameter>
  <!-- This is the key that will be passed to the -->
  <!-- to the ObjectMarshallerDispatcher service  -->
  <!-- The key is used to determine which         -->
  <!-- marshaller to use.                         -->
  <action-parameter>
    <action-parameter-name>
      marshalKey
    </action-parameter-name>
    <display-name-resource>
      sendObjectAsXML.marshalKey.displayName
    </display-name-resource>
    <action-parameter-class>
      java.lang.String
    </action-parameter-class>
    <required>
      false
    </required>
    <description-resource>
      sendObjectAsXML.marshalKey.description
    </description-resource>
  </action-parameter>
</action>
```

このコードでは、ATG Control Center 内で「Send Object As XML via SOAP」として表示される、アクションとシナリオ・サーバーを登録します。このアクションは、ATG Control Center 内でシナリオを作成するときに指定される 2 つのパラメータを受け取ります。

1 番目のパラメータは `marshalObject` で、これは必須です。このパラメータは、XML にシリアル化されるオブジェクトです。通常、このオブジェクトはソース・イベントから取得されます。

2 番目のパラメータは `marshalKey` で、これは省略可能です。これはユーザーから提供されるキー・オブジェクトであり、`ObjectMarshallerDispatcher` に渡されて、マーシャリングを実行する `ObjectMarshaller` プロセッサを決定します。

詳細は、『ATG Web Commerce Installation and Configuration Guide』の「*サイトのパフォーマンスのモニタリング*」の*構成レポータの使用*に関する項を参照してください。

`marshalObject` および `marshalKey` の構成に加えて、`SendObjectAsXML` アクションの動作も変更できます。そのプロパティの大部分は、`atg.projects.b2bstore.soap.SOAPResources` ファイルを介して構成できます。

SOAPResources ファイルには、次のリソース文字列があります。

- SimpleSOAPClient は、SOAP 経由でメッセージを送信するときに使用する、SimpleSOAPClient の Nucleus 位置を決定します。デフォルトは /atg/projects/b2bstore/soap/SimpleSOAPClient です。
- MarshalServicePath は、オブジェクトを XML ドキュメントにマーシャルするときに使用する、ObjectMarshallerDispatcherService の Nucleus 位置を決定します。デフォルトは/atg/dynamo/service/xml/ObjectMarshallerDispatcher です。
- SOAPParameterName は、SOAP ペイロードに埋め込まれるパラメータの名前を決定します。デフォルトは xmlDocument です。

シナリオのカスタム・アクションの作成の詳細は、『ATG Web Commerce Personalization Programming Guide』の*カスタム・イベントおよびアクションのシナリオへの追加*に関する項を参照してください。

Motorprise における SOAP のサーバー要素

ストア内の SOAP サーバーは、サーバー自体およびそのサーバーに登録される SOAP サービスという、2 つの要素からなります。

SOAP サーバー

ストア内の SOAP サーバーは、受信する SOAP メッセージをリスニングし、それを特定の宛先にルーティングします。サーバー（SOAP ルーターと呼ばれることもあります）は、実際には、メッセージを特定の SOAP サービスにルーティングする Java サーブレットです。

最初に、SOAP サーバーとしての役割を果たす次の Nucleus コンポーネントを作成しました: /atg/projects/b2bstore/soap/RPCRouterServlet。このコンポーネントのプロパティ・ファイルは、次のようになります。

```
$class=atg.server.soap.DynamoRPCRouterServlet
$scope=global
serviceManager=ServiceManager
RPCRouter=RPCRouter
```

RPCRouterServlet コンポーネントには、構成済の ServiceManager および RPCRouter コンポーネントが必要です。

ServiceManager コンポーネントは、特定の SOAPRouter で使用可能な、公開済のすべてのサービスのリストを保持しています。この ServiceManager に登録されている単一のサービスが、次の項に記載されている DOMWriterService です。

/atg/projects/b2bstore/soap/ServiceManager のプロパティ・ファイルを、次に示します。

```
$class=atg.apache.soap.server.ServiceManager
$scope=global
deployedServicesFile={atg.dynamo.server.home}/data/motorprise/
D6DeployedServices.ds
```

Motorprise 固有の RPCRouter コンポーネントが/atg/projects/b2bstore/soap/RPCRouter に作成されています。RPCRouter コンポーネントは、特定の RPC 要求を特定のサービスにルーティングします。このコンポーネントは、HTTP や E メールなど、要求のトランスポートの方法にかかわらず、受信する SOAP RPC 要求を処理できます。

RPCRouterServlet は HTTP 経由で要求を受信し、SOAP メッセージを抽出し、それを RPCRouter コンポーネントに転送します。RPCRouter コンポーネントは SOAP メッセージを調べて、メッセージの宛先のサービスを決定します。

最後に、特定の URL に送信されるメッセージが RPCRouterServlet に確実にルーティングされる必要があります。/Dynamo/solutions/B2BStore/soap（SOAP クライアント内で定義）に送信される HTTP 要求が、RPCRouterServlet に自動的に転送されることが必要です。/atg/dynamo/Configuration の dispatcherServiceMap プロパティを使用してこれらの要求を転送するために、<ATG11dir>/Motorprise/config/atg/dynamo/にある Configuration.properties ファイルに次の内容が追加されています。

```
dispatcherServiceMap+=\
     /Dynamo/solutions/B2BStore/soap=/atg/projects/b2bstore/soap/
       RPCRouterServlet
```

このため、SOAP クライアントが/Dynamo/solutions/B2BStore/soap の場所にメッセージを送信すると、メッセージが SOAP サーバーに転送されます。

SOAP サービス

SOAP サービスには、起動されて SOAP メッセージのペイロードを処理する、特定のメソッド・コールがあります。Motorprise では、XML ドキュメントを Dynamo コンソールに印刷するサービスが必要です。指定された次の XML ドキュメントを印刷するクラスが作成されています: atg.projects.b2bstore.soap.DOMWriterService。また、このクラス（/atg/projects/b2bstore/soap/DOMWriterService）のインスタンスが次のプロパティとともに作成されています。

```
$class=atg.projects.b2bstore.soap.DOMWriterService
$scope=global
RPCRouterServlet=/atg/projects/b2bstore/soap/RPCRouterServlet
methods=receiveDocument
serviceId=urn\:motorprise-display-xml
```

/atg/projects/b2bstore/soap/DOMWriterService クラスは、SOAP サーバーに SOAP サービスを登録する doStartService というメソッドを持ちます。DOMWriterService コンポーネントが開始すると、doStartService メソッドが、DOMWriterService コンポーネントの構成済プロパティ RPCRouterServlet を通じて SOAP サーバーを決定します。doStartService メソッドは SOAP サーバーに SOAP サービスを登録するため、サーバーは、service urn:motorprise-display-xml の要求と receiveDocument のメソッド・コールを、/atg/projects/b2bstore/soap/DOMWriterService に登録されている SOAP サービスへルーティングします。

11 マーチャンダイジング

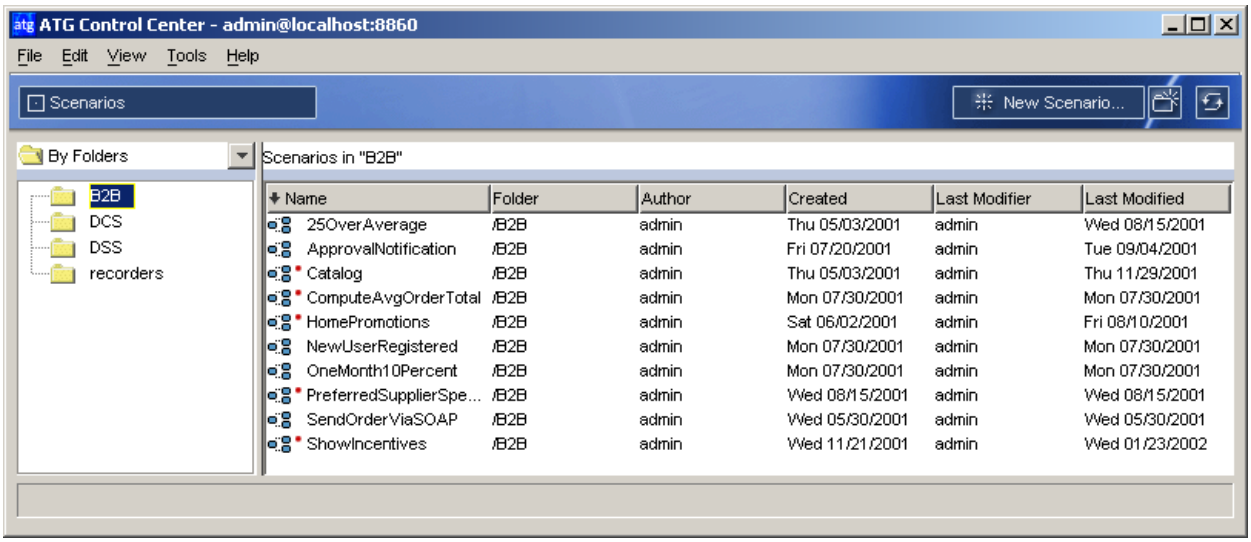
Motorprise では Oracle Commerce Platform のシナリオ機能を使用して、顧客にパーソナライズされたマーチャンダイジングを提供します。Motorprise 固有のシナリオを作成しましたが、それに加えて、Motorprise では Core Commerce のシナリオも利用します。Motorprise 内にカスタム条件も作成されています。

この章は次の項から構成されています。

Motorprise シナリオ

Motorprise 用に作成されたシナリオは、ATG Control Center の「シナリオ」→「シナリオ」セクションの B2B フォルダで表示できます。

デフォルトでは、すべての Motorprise シナリオが有効になっているわけではありません（赤のドットが有効なシナリオを表します）。E メールを送信するシナリオは無効になっており、E メール・サーバーが設定されると有効にできます。詳細は、このマニュアルの「概要」の「E メールの設定」の項を参照してください。



ATG Control Center における Motorprise シナリオ

25OverAverage

このシナリオは、発行されたオーダーの合計金額の増加の割合がこれまでのオーダーの平均金額の 25% を超えるかどうかをチェックします。超える場合、ユーザーは次回のオーダーで 10%の販促を受けます。このシナリオは、`ComputeAvgOrderTotal` シナリオでバイヤーのプロファイルに記録される、オーダー数と個人の平均オーダー価格の値に依存します。詳細は、このシナリオに関する次の説明を参照してください。

このシナリオの最初の手順では、オーダーが完了しているかどうかをチェックします。オーダーが完了している場合、そのサブタイプは「オーダー終了」に変わります。オーダーが発行されたかどうかではなく、完了しているかどうかをチェックするのは、後で取り消されるオーダーに販促を与えることを回避するためです。次の手順では、このバイヤーが 5 件のオーダーを発行していて、平均オーダー合計額が確立されているかどうかをチェックします。

次に、オーダーの増加の割合がバイヤーの平均オーダー合計額の 25%を超えるかどうかをチェックします。これは、Motorprise 用に作成された Order Comparison というカスタム条件です。異なるパーセンテージを入力して調べることもできます。オーダーの増加の割合が平均オーダー合計額の 25%を超える場合、バイヤーは次回のオーダーで 10%オフの販促を受けます。販促に関する E メールも受信します。Motorprise には英語とドイツ語の顧客がいるため、該当の E メール・テンプレートを選択する前に顧客のロケールをチェックして、適切な言語で E メールを送信します。

Motorprise 内のユーザー・プロファイルには、発行されたオーダー数を示すプロパティ `numOfOrders`、およびユーザーのすべてのオーダーの平均合計額を追跡するプロパティ `avgOrderAmt` が追加されています。新規オーダーを発行すると、この 2 つのプロパティが適切な値で更新されます。

ユーザーの平均オーダー合計額を計算するために、新規シナリオ・アクション Compute Average Order Total が作成されています。このアクションはオーダー合計額を入力として受け取り、`avgOrderAmt` および `numOfOrders` から平均オーダー合計額を計算します。

このアクションに基づき、`OrderSumitted` をイベントとし、オーダー金額をアクションへの入力として、シナリオが作成されます。また、`numOfOrders` は、オーダーが作成されるたびに 1 ずつ増加します。オーダーが発行されると、このアクションがオーダー金額を入力として起動され、新しい平均オーダー合計額が計算されて、その値が `avgOrgAmt` に格納されます。

シナリオ・アクションの作成の詳細は、『ATG Web Commerce Personalization Programming Guide』を参照してください。

ApprovalNotification

承認が必要なオーダーについて通知する必要がある状況には、次の 3 つの場合があります。

- 承認が必要なオーダーが発行されると、承認者が通知されます。
- 承認者がオーダーを承認して発行すると、バイヤーが通知されます。
- 承認者がオーダーを否認すると、バイヤーが通知されます。

承認が必要なオーダーが発行されると、Approval Required イベントが送信され、承認者に E メールが送信されます。最初にユーザーのロケールをチェックして、適切な E メールを送信します。承認者がオーダーを承認または否認すると、バイヤーに E メールが送信され、そのオーダーの状態が伝えられます。

これは、ドイツ語の分岐を追加するために、Motorprise 内で上書きする b2bcommerce シナリオです。

Catalog

匿名セッションが開始すると、このシナリオによってベース・カタログが割り当てられます。これを使用することで、親組織と Motorprise との契約に基づいて、既知のユーザーにカタログを提供できます。

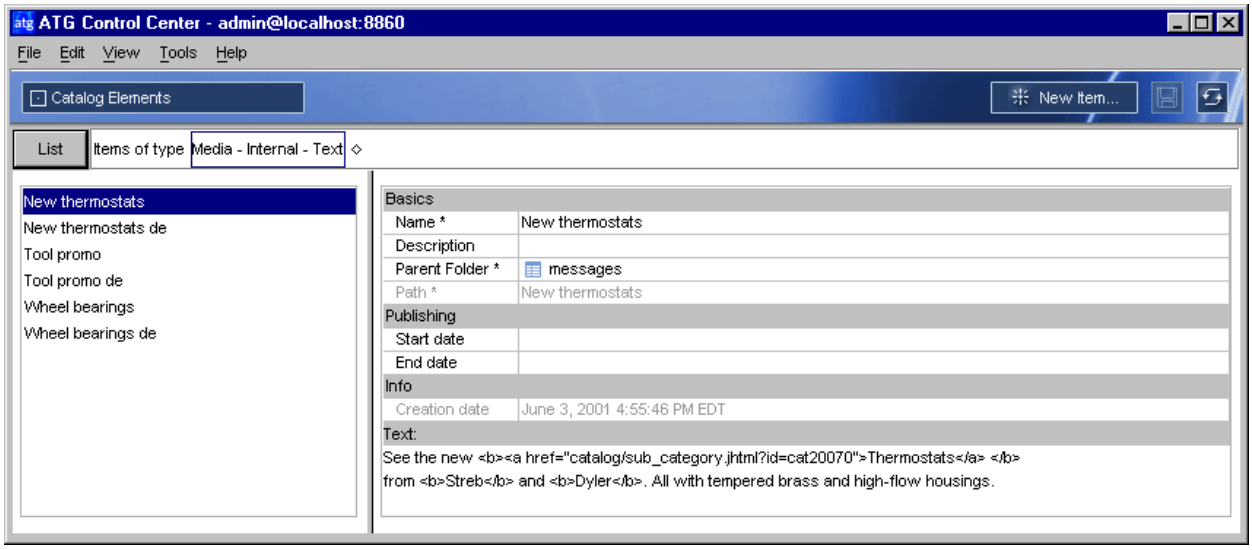
ComputeAvgOrderTotal

このシナリオは、25OverAverage シナリオで使用される情報を記録します。オーダーが発行されると、バイヤーのオーダー数が 1 件増加し、カスタム・アクションを使用してバイヤーの平均オーダー合計額が計算されます。

HomePromotions

ユーザーが Motorprise ホーム・ページにログインすると、特定の販促が表示されます。

販促は 3 種類あり、それぞれがテキスト・スロットおよびイメージ・スロットという 2 つのスロットで構成されています。ホーム・ページ上でこれらのスロットを使用して、ユーザーにテキスト・メッセージとイメージが配信されます。テキスト・メッセージは、データベースに格納されている内部メッセージです。メッセージを確認するには、「カタログ管理」→「カタログ要素」に進み、タイプ `media-internal-text` の項目を表示します。



HomePromotions シナリオで使用されるテキスト・メディア

イメージは各製品に格納されている `small images` で、カタログ内の対応する製品で表示できます。

各スロットとも、ユーザーがホーム・ページを訪問すると、テキスト・メッセージがテキスト・スロットに追加され、対応するイメージがイメージ・スロットに追加されます。

NewUserRegistered

サイト管理者が `admin/new_user.jsp` または `admin/create_multiple_users.jsp` を使用して新規ユーザーを登録すると、ユーザー名とサイトへのリンクが含まれる E メールがユーザーに送信されます。

OneMonth10Percent

顧客がオーダーを発行すると、30 日間待機して、ユーザーがオーダーを発行したかどうかをチェックするシナリオ・インスタンスが作成されます。オーダーが発行されなかった場合は、10%オフの販促を通知する E メールが送信されます。

PreferredSupplierSpecials

ユーザーが Motorprise ホーム・ページを訪問すると、「Preferred Vendor Specials」セクションに優先ベンダーの特定の製品が表示されます。

Motorprise で購買組織を設定するときに、顧客は会社が優先して購入するベンダーまたはサプライヤを指定します。これは組織プロファイルに入力され、各バイヤーがサイトにアクセスするときに、Motorprise はその会社の優先ベンダーを識別できます。このため、Motorprise では、ベンダーに基づいてバイヤーの興味を引く可能性が高い製品の販促を行うことができます。このような製品がホーム・ページでバイヤーに対して表示されます。

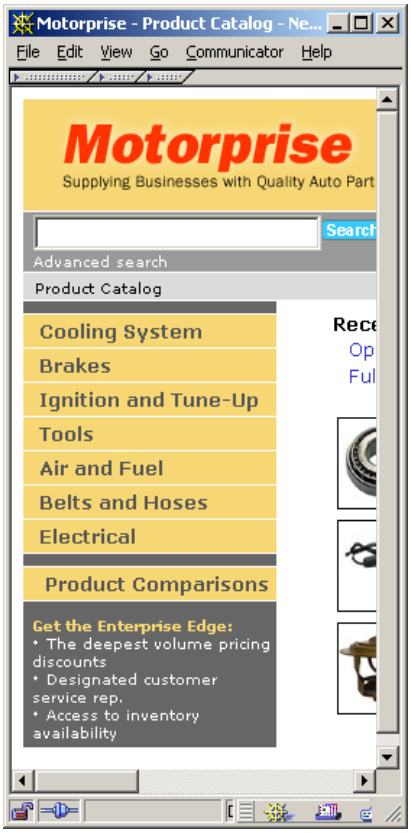
バイヤーが英語またはドイツ語のホーム・ページを訪問すると、最初に、スロット内に存在する場合がある項目が削除されます。これは、ユーザーがログイン前に匿名ユーザーとしてサイトを参照していた場合であり、対象製品ではなく、匿名ユーザーに表示されるデフォルト製品がスロットに入っている可能性があります。次に、優先ベンダー・プロパティをチェックして、ベンダーごとに販促を行う製品を表示します。ベンダー分岐の最後には Otherwise 分岐があり、匿名ユーザー、および親組織に優先ベンダーが定義されていないユーザーに対して製品のデフォルト・リストが表示されます。

SendOrderViaSoap

このシナリオは SOAP 経由でオーダーを送信するため、通販代行会社や Motorprise パートナのシステムなど、他のシステムがオーダーを受信して読み取ることができます。このシナリオの詳細は、このマニュアルの「SOAP サポート」を参照してください。

ShowIncentives

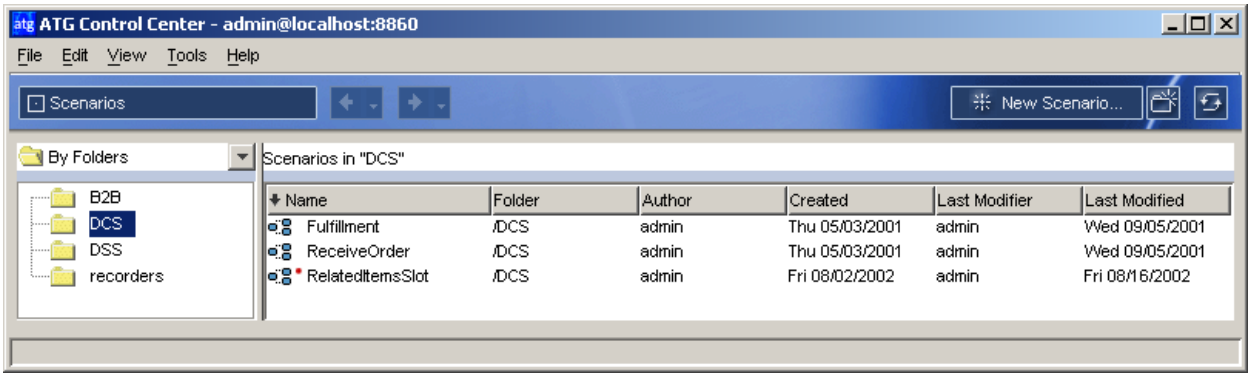
このシナリオは、ユーザーが匿名、NDAP ユーザーまたは USMW ユーザーかどうかに応じて、カタログ・カテゴリの下にそれぞれ異なるインセンティブ・コンテンツを表示します。



NDAP ユーザーには企業顧客になることを奨励するインセンティブが表示される。

Core Commerce シナリオ

Motorprise では、ATG Control Center の「シナリオ」→「シナリオ」の DCS フォルダに表示されるコア・シナリオを使用します。



Motorprise で使用される Core Commerce シナリオ

デフォルトでは、RelatedItemsSlot シナリオが有効です。場合によっては、E メールを送信する Fulfillment シナリオおよび ReceiveOrder シナリオが有効になっています。詳細は、このマニュアルの「概要」の「*E メールの設定*」の項を参照してください。

Fulfillment

Fulfillment シナリオは、次のいずれかのイベントが発生すると、顧客に適切な E メールを送信します。

- オーダーが出荷される場合
- オーダーに使用不可の品目がある場合
- オーダーが取り消される場合
- オーダーから品目が削除される場合
- 支払情報が変更される場合

ReceiveOrder

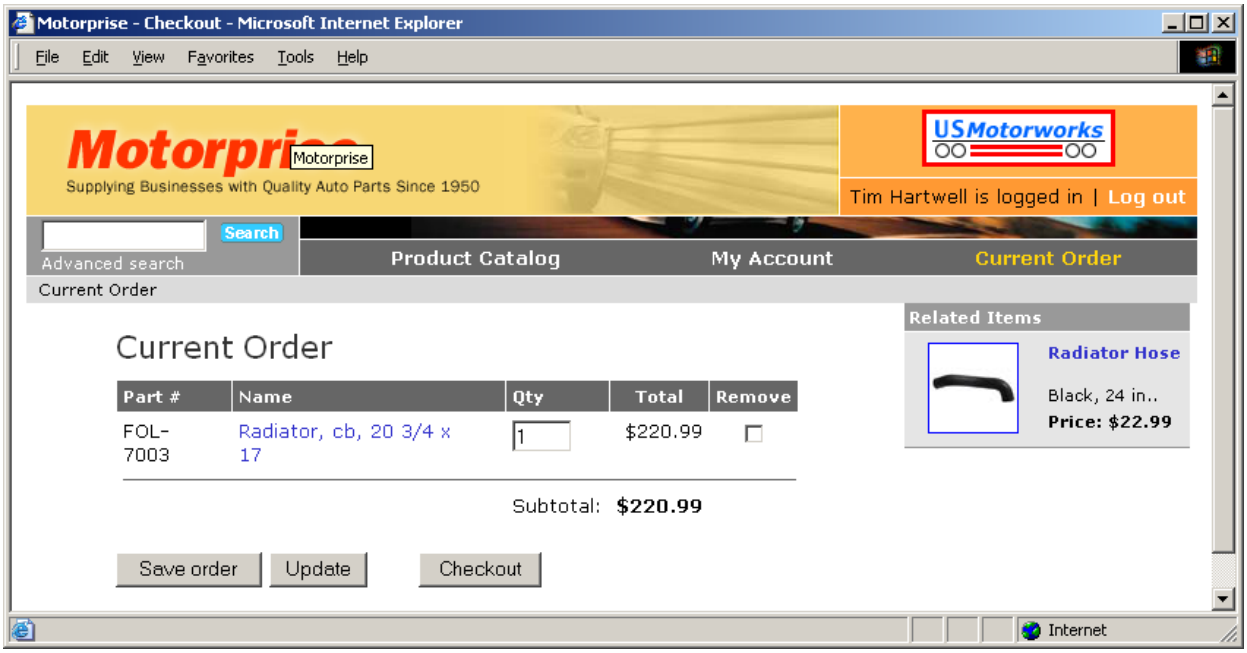
ReceiveOrder シナリオはオーダーが発行されるのを待ち、オーダーを受け取ったことを通知する E メールを顧客に送信します。

Fulfillment シナリオおよび ReceiveOrder シナリオの詳細は、『ATG Web Commerce Programming Guide』を参照してください。

RelatedItemsSlot

Motorprise は RelatedItemsSlot シナリオを使用して、顧客の現在のショッピング・カート内の製品に関連する製品の、抱合せ販売を行います。シナリオでは、ショッピング・カート・ページ `cart.jsp` で使用される RelatedItemsOfCart という名前のスロットのコンテンツを判断して、顧客のオーダー内の製品に関連する品目を表示します。

たとえば、Tim Hartwell などの顧客がオーダーにラジエータを入れると、スロットにはラジエータ・ホースが表示されます。各製品には `relatedProducts` プロパティがあり、これを使用して、オーダー・ページ上のスロットに表示する製品が決定されます。



ショッピング・カート・ページで使用される RelatedItemsOfCart スロット

RelatedItemsSlot シナリオの詳細は、『ATG Web Commerce ストア設定コマース・ガイド』の「*シナリオ内の Commerce 要素の使用*」の*クロスセルおよびアップセル製品へのシナリオの使用*に関する項を参照してください。

Motorprise におけるカスタム条件の作成

Motorprise では、顧客の現在のオーダー価格が平均オーダー価格の 25%以上増しの場合、顧客に対して特別な販促を提供します。この要件を処理するために、Scenarios モジュール内にカスタム条件が実装され、こうした条件の作成および編集用にカスタマイズされたユーザー・インタフェースを提供するために、ATG Control Center のシナリオ・エディタ内の式構文が拡張されています。

『ATG Web Commerce Personalization Programming Guide』の「*カスタム・イベント、アクションおよび条件のシナリオへの追加*」には、カスタム条件および式構文の作成方法が記載されています。

カスタム条件は、クラス `atg.projects.b2bstore.scenario.PercentageComparisonFilter` を使用して実装されており、これにより、Scenarios モジュールの標準クラス `ExpressionFilter` が拡張されます。

`PercentageComparisonFilter` は、次のタイプの条件をサポートします。

```
testValue [is at least / is at most]
   percentage percent [greater than / less than / of] baseValue
```

ここで、`testValue` および `baseValue` は数式、`percentage` はリテラル・パーセント値のプレースホルダです。

たとえば、Motorprise では、この条件クラスを「Order's price is at least 25% greater than person's average order price」というタイプの条件で使用します。この例の場合、`testValue` が「order's price」、`baseValue` が「person's average order price」、`percentage` が「25」です。

このカスタム条件は、起動時に 5 つのオペランドを受け取ることが期待されます。

- Operand 1 はテスト値で、数値または数式です。
- Operand 2 は、比較タイプ（少なくとも/最大で）を指定します。文字列定数は、このオペランドで可能な各値を表します。
- Operand 3 は条件内のリテラル・パーセント値で、数値です。
- Operand 4 は、テスト値に対する条件（より大きなパーセンテージ/未満のパーセンテージ/基準値のパーセンテージ）を指定します。比較タイプと同様、文字列定数は可能な各値を表します。
- Operand 5 は基準値で、やはり数値または数式です。

scenarioManager.xml でのカスタム条件の登録時にオペランドの数とタイプが指定されています。

```
<condition-registry>
  <condition>
    <condition-name>orderPricePercentComparison</condition-name>
    <filter-class>atg.projects.b2bstore.scenario.
      PercentageComparisonFilter</filter-class>
    <resource-bundle>atg.projects.b2bstore.scenario.UserResources
    </resource-bundle>
    <display-name-resource>orderPricePercentComparison.displayName
    </display-name-resource>
    <description-resource>orderPricePercentComparison.description
    </description-resource>

<!--
    The action-parameter tags specify positional parameters that will
    be passed to the filter class's initialize() method.
-->

    <action-parameter>
      <action-parameter-name>
        testValue
      </action-parameter-name>
      <display-name-resource>
        orderPricePercentComparison.testValue.displayName
      </display-name-resource>
      <action-parameter-class>
        java.lang.Number
      </action-parameter-class>
      <required>
        true
      </required>
      <description-resource>
        orderPricePercentComparison.testValue.description
      </description-resource>
    </action-parameter>

    <action-parameter>
```

```
  <action-parameter-name>
    comparisonType
  </action-parameter-name>
  <display-name-resource>
    orderPricePercentComparison.comparisonType.displayName
  </display-name-resource>
  <action-parameter-class>
    java.lang.String
  </action-parameter-class>
  <required>
    true
  </required>
  <description-resource>
    orderPricePercentComparison.comparisonType.description
  </description-resource>
</action-parameter>

<action-parameter>
  <action-parameter-name>
    percentage
  </action-parameter-name>
  <display-name-resource>
    orderPricePercentComparison.percentage.displayName
  </display-name-resource>
  <action-parameter-class>
    java.lang.Integer
  </action-parameter-class>
  <required>
    true
  </required>
  <description-resource>
    orderPricePercentComparison.percentage.description
  </description-resource>
</action-parameter>

<action-parameter>
  <action-parameter-name>
    percentageType
  </action-parameter-name>
  <display-name-resource>
    orderPricePercentComparison.percentageType.displayName
  </display-name-resource>
  <action-parameter-class>
    java.lang.String
  </action-parameter-class>
  <required>
    true
  </required>
  <description-resource>
```

```
        orderPricePercentComparison.percentageType.description
      </description-resource>
    </action-parameter>

    <action-parameter>
      <action-parameter-name>
        baseValue
      </action-parameter-name>
      <display-name-resource>
        orderPricePercentComparison.baseValue.displayName
      </display-name-resource>
      <action-parameter-class>
        java.lang.Number
      </action-parameter-class>
      <required>
        true
      </required>
      <description-resource>
        orderPricePercentComparison.baseValue.description
      </description-resource>
    </action-parameter>

  </condition>
</condition-registry>
```

ATG Control Center シナリオ・エディタでこの条件のインスタンスを表示および編集する方法を制御するために、カスタム構文拡張も作成されています。式構文は、通常、『ATG Web Commerce Personalization Programming Guide』の「*カスタム・イベント、アクションおよび条件のシナリオへの追加*」の「構文エディタの拡張」に記載されているパターンに従いますが、次の 2 つの点については追加の説明が必要です。

第 1 に、標準の<sequence>タグを使用して条件構文を開始するかわりに、次の DCS <required-order-sequence>タグが使用されています。

```
<required-order-sequence id="condition-orderPricePercentComparison">
```

このタグを使用することで、DCS オーダー・イベントをすでに含むシナリオ・セグメント内にのみカスタム条件が存在することが、シナリオ・エディタに対して指定されます。

それ以外の場合、「Order's price」のような式を条件の一部として評価する方法はありません。

第 2 に、条件の第 1 オペランド（テスト値）を事前に定義した 4 つの値の 1 つに制限するために、カスタム構文が使用されています。PercentageComparisonFilter クラスはテスト値として任意の数式を受け取ることができますが、ビジネス・ユーザーが次の値の 1 つを選択するように制限されています。

- オーダー価格
- 割引前のオーダー価格
- オーダー税
- オーダー送料

これは、<choice>タグを第 1 オペランドとして使用することで実現されています。choice 内の各トークンは 4 つの可能な値の 1 つを表します。

```
<choice>
  <token>
    <description>Order price</description>
    <xml-template>
      <event-property>
        <property-name>order</property-name>
        <property-name>priceInfo</property-name>
        <property-name>total</property-name>
      </event-property>
    </xml-template>
  </token>

  <token>
    <description>Order price before discounts</description>
    <xml-template>
      <event-property>
        <property-name>order</property-name>
        <property-name>priceInfo</property-name>
        <property-name>rawSubTotal</property-name>
      </event-property>
    </xml-template>
  </token>

  <token>
    <description>Order tax</description>
    <xml-template>
      <event-property>
        <property-name>order</property-name>
        <property-name>priceInfo</property-name>
        <property-name>tax</property-name>
      </event-property>
    </xml-template>
  </token>

  <token>
    <description>Order shipping cost</description>
    <xml-template>
      <event-property>
        <property-name>order</property-name>
        <property-name>priceInfo</property-name>
        <property-name>shipping</property-name>
      </event-property>
    </xml-template>
  </token>
</choice>
```

各トークンには、シナリオ・エディタに表示される摘要と、SDL（シナリオ定義言語）を使用した、対応するイベント・プロパティ用の XML テンプレートが含まれています。各イベント・プロパティ定義により、DCS オー

ダー・イベントの 1 つのプロパティが指定されます。たとえば、イベント・プロパティ定義は、SDL 形式のイベントの order.priceInfo.total サブプロパティです。

```
<event-property>
  <property-name>order</property-name>
  <property-name>priceInfo</property-name>
  <property-name>total</property-name>
</event-property>
```

12 ATG Control Center ユーザーの作成および権限の設定

Motorprise には、サイト・コンテンツの様々なセクションにアクセスする必要がある様々なビジネス・ユーザーがいます。これらのユーザーが ATG Control Center を使用して様々な管理タスクを実行できるように、ロールが設定されています。あるユーザーはカタログと価格表を管理します。組織、契約などを作成することによって、新規顧客アカウントを設定するユーザーもいます。

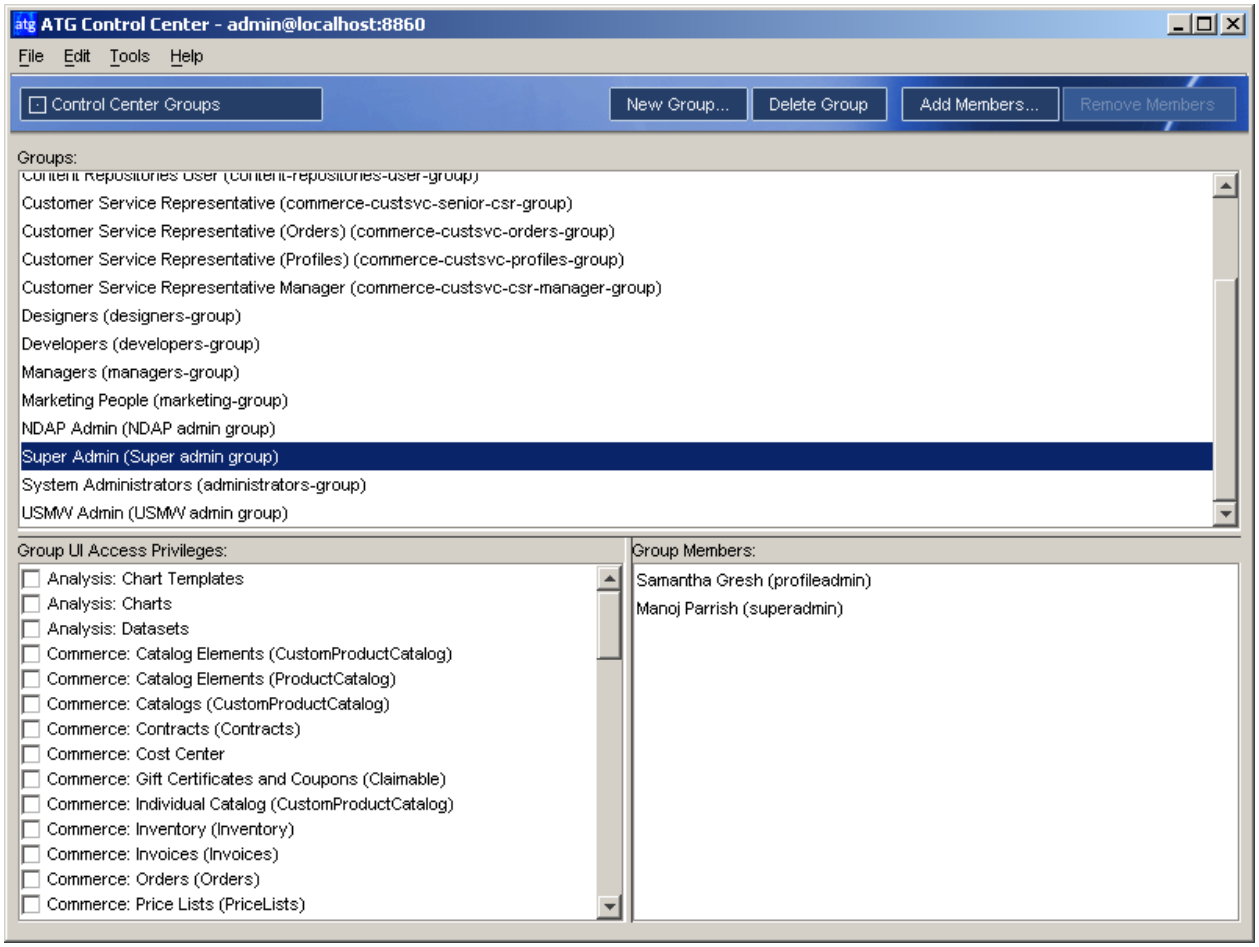
これらのユーザーの作成およびアクセス権限の定義は、管理者として ATG Control Center にログインすることで行われました（ユーザー名およびパスワードに admin を使用）。

Motorprise ATG Control Center ユーザーは次のように設定されています。

- 5 人のユーザーの作成: Manoj、Eddie、Sam、Manny、Samantha
- 5 つのロールの作成: Superadmin、Content QA、NDAP Account Manager、USMW Account Manager、Profileadmin
- 特定のユーザーへのこれらのロールの割当て
- 様々なロールのアクセスの設定

ATG Control Center におけるグループの作成

Motorprise グループの作成には、「個人および組織」タスク・メニューの「Control Center グループ」セクションが使用されました。「新規グループ」ボタンをクリックして各グループの名前および摘要を入力するのみです。



ATG Control Center の「個人および組織」セクション内の Control Center グループ。

4 つのグループが作成されています。

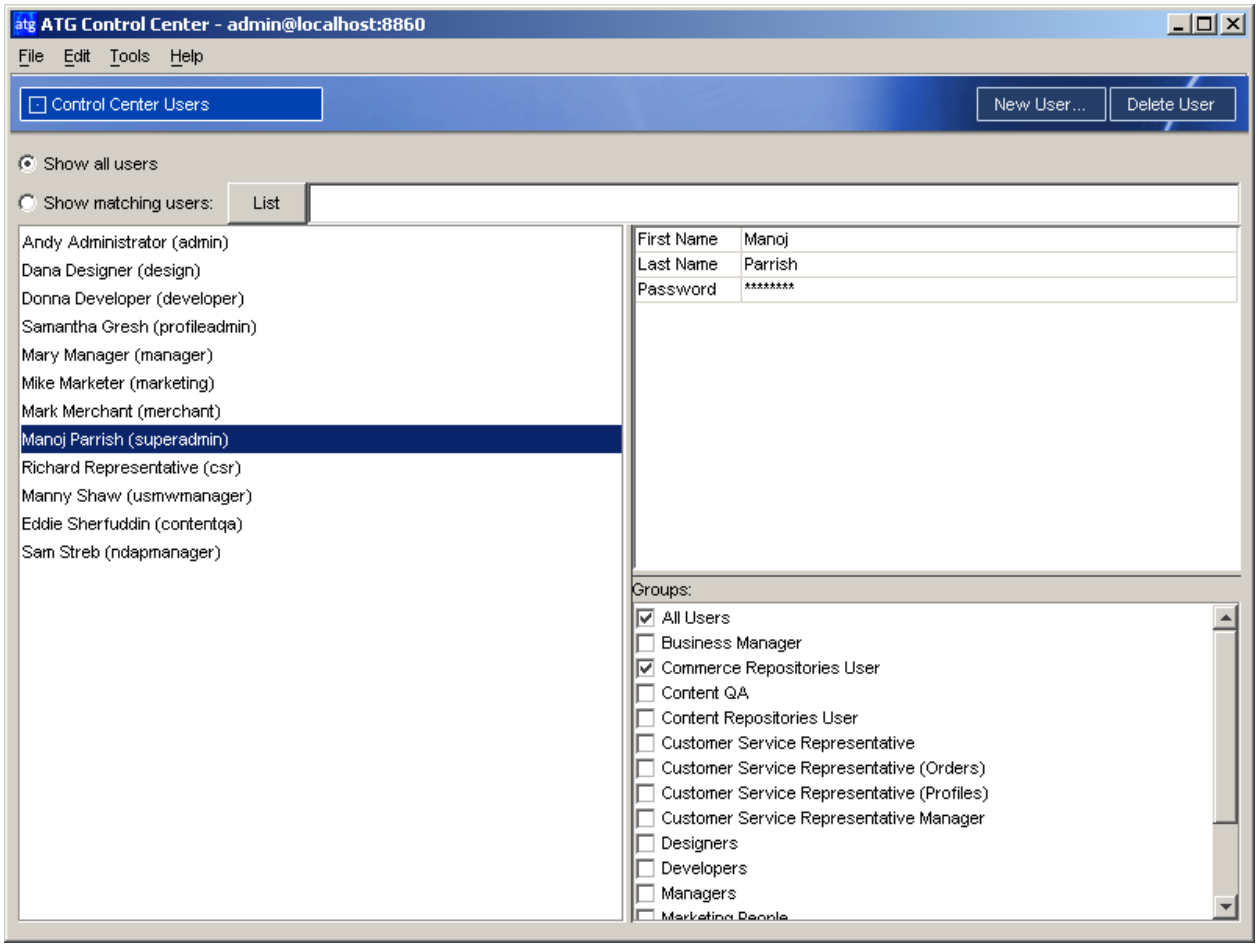
- Content QA
- NDAP Admin
- Super Admin
- USMW Admin

グループごとに特定のグループ UI アクセス権限も指定できます。たとえば、「Commerce Repositories Users」がアクセスできるのは ATG Control Center 内のコマース・タスク・メニューのみです。

ATG Control Center におけるユーザーの作成

新規 Motorprise ATG Control Center ユーザーの作成は、「個人および組織」タスク・メニュー→「Control Center ユーザー」タブで、**「新規ユーザー」**ボタンをクリックし、名前とパスワードの情報を入力することで行われました。

また、各ユーザーが適切な権限のグループに追加されました。たとえば、Manoj Parrish は「Super Admin」および「Commerce Repositories User」になっています。(デフォルトでは、彼は「All Users」グループのメンバーです)。



ユーザーManoj Parrish がグループ「All Users」、「Commerce Repositories」および「Super Admin」に割り当てられる。

ユーザー	ロール	グループ
Manoj Parrish	Motorprise Superadmin	All Users Commerce Repositories User Super Admin
Eddie Sherfuddin	Motorprise Content QA	All Users Commerce Repositories User Content QA
Sam Streb	Motorprise アカウント・マネージャ - NDAP	All Users Commerce Repositories User NDAP Admin

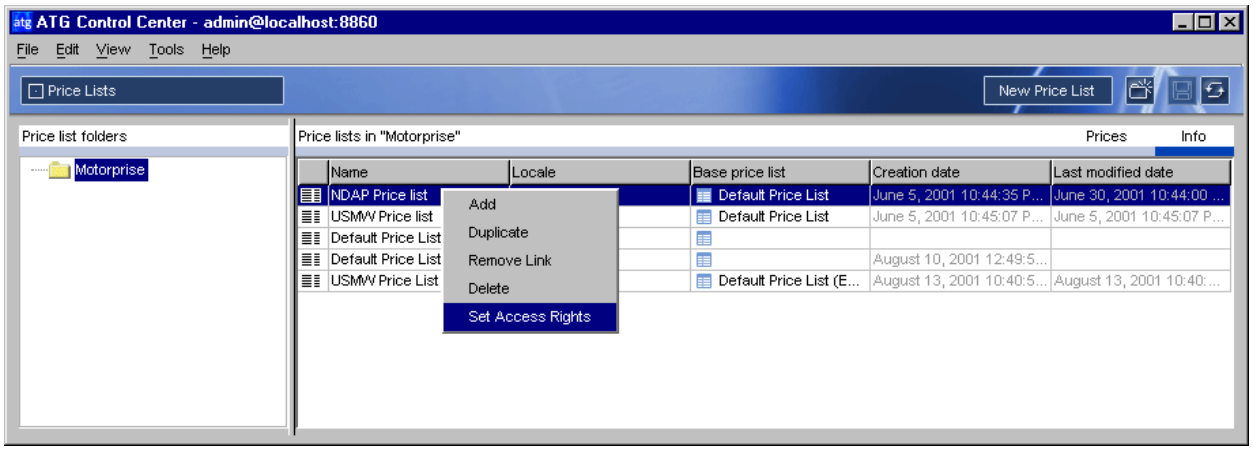
Manny Shaw	Motorprise アカウント・マネージャ - USMW	All Users Commerce Repositories User USMW Admin
Samantha Gresh	Motorprise Profileadmin	All Users Commerce Repositories User Content Repositories User Super Admin

価格表へのアクセス権限の割当て

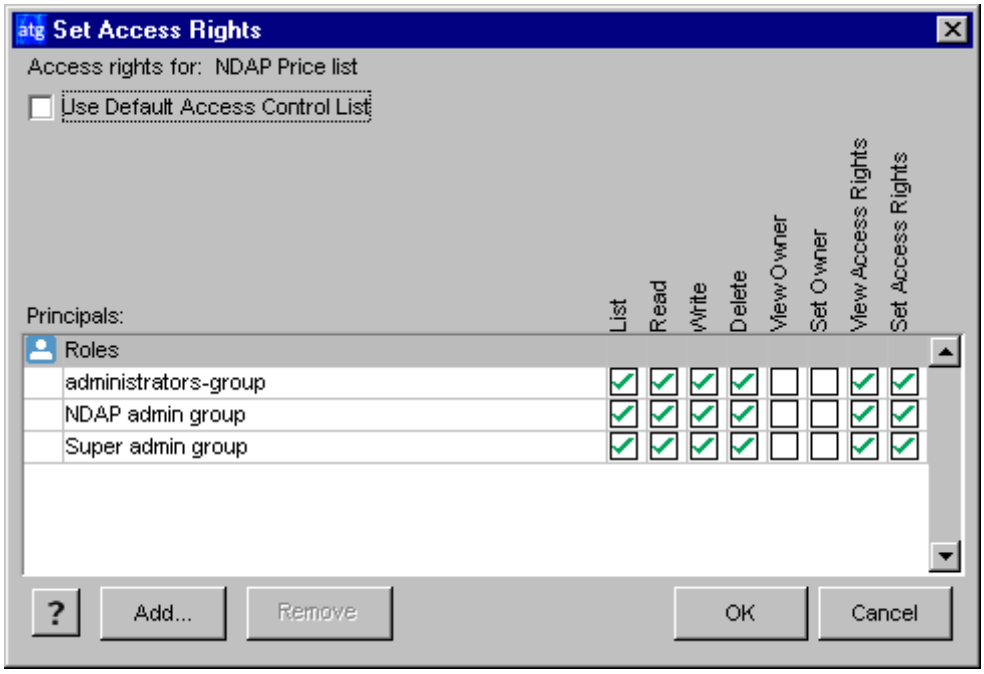
グループを作成してユーザーを割り当てた後、特定のユーザーに、特定の価格表へのアクセスを許可する必要があります。これらのアクセス権限は、その特定の価格表内のすべての価格および複合価格で継承されます。

たとえば、NDAP 価格表における Content QA 管理者のアクセス権限を設定するには、次の操作を行います。

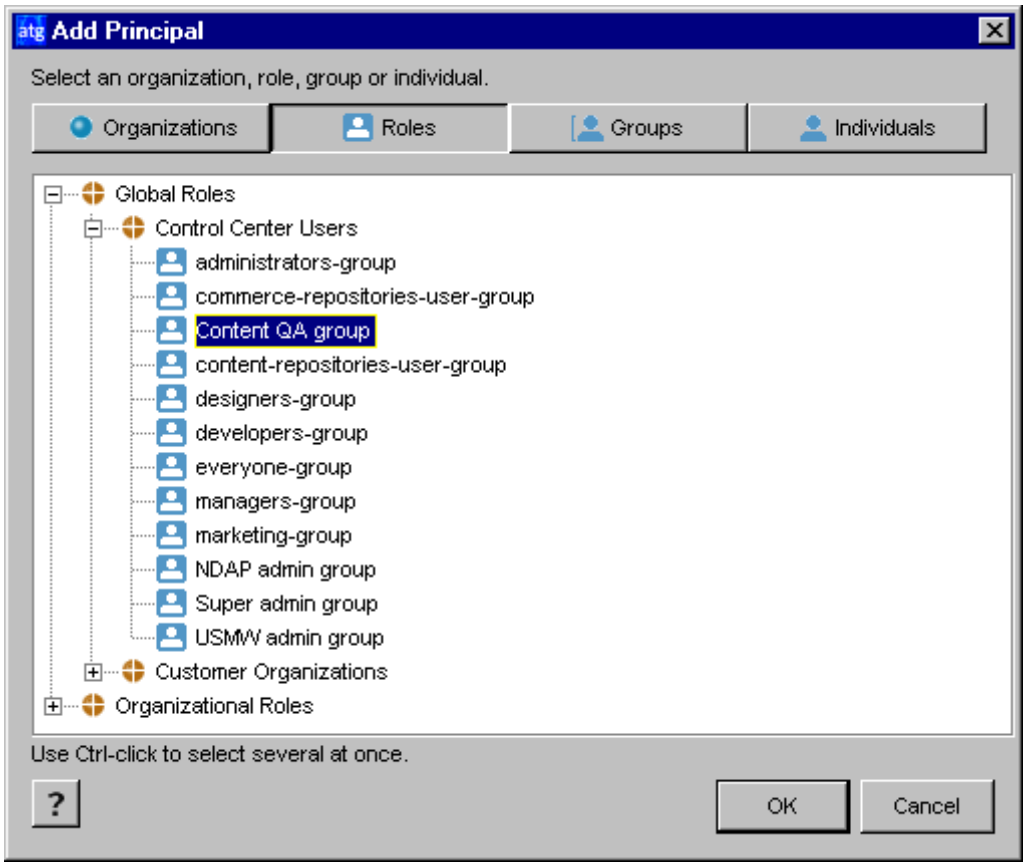
1. ATG Control Center の「価格設定」→「価格表」タスク・セクションで、右上にある「情報」タブを選択して Motorprise 価格表（NDAP PriceList、USMW PriceList、USMW PriceList（EUR）、Default Price List および Default Price List（EUR））を表示します。



2. 「NDAP PriceList」を選択して右クリックし、「アクセス権限の設定」を選択します。



3. 「アクセス権限の設定」ダイアログ・ボックスで、「**追加**」ボタンをクリックします。



4. 「ロール」タブを選択します。
5. 「グローバル・ロール」を展開し、「Control Center ユーザー」を展開して、これまでに作成されたすべての Control Center グループを表示します。
6. 「Content QA group」を選択し、「OK」をクリックします。Content QA が「アクセス権限」ダイアログ・ボックスに追加されます。
7. アクセス権限をリストしている列で、「リスト」ボックスおよび「読取り」ボックスを選択します。

アクセス権限の詳細は、『ATG Web Commerce Personalization Programming Guide』を参照してください。

カタログへのアクセス権限の割当て

価格表の保護に加えて、Motorprise ではセキュアなカタログも作成されています。カタログを保護することで、特定のユーザーのみがカタログを表示および編集できます。たとえば、NDAP アカウント・マネージャには NDAP カタログの読取りおよび書込みのアクセス権はありますが、USMW カタログへのアクセス権限はありません。

各ユーザーのアクセス権限の動作方法については、次の「販売組織としての Motorprise の操作」を参照してください。

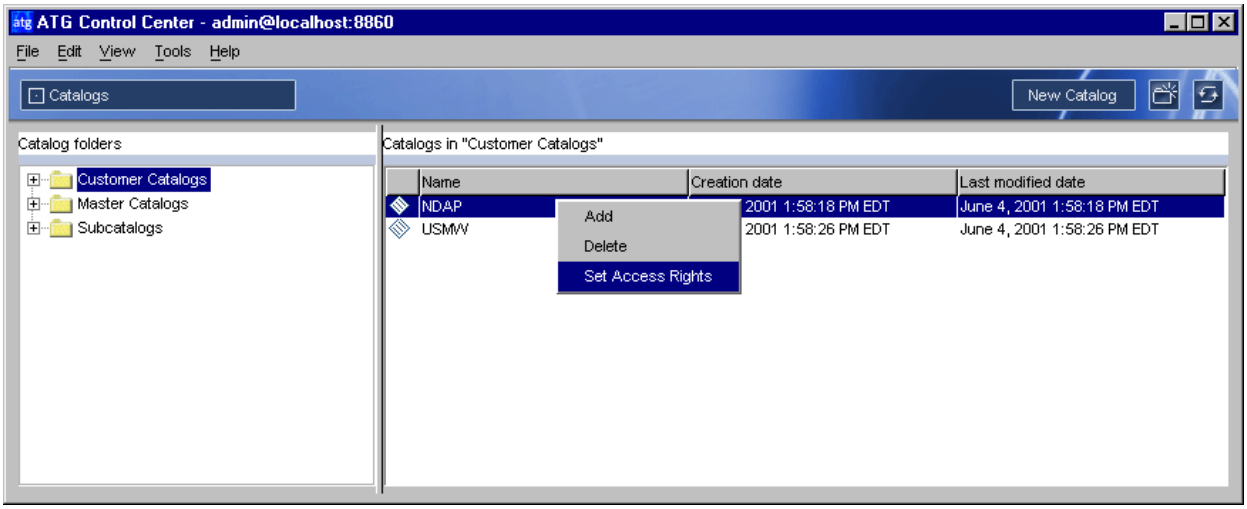
ユーザーが価格表やカタログの表示または編集を試みると、セキュリティ・システムがオブジェクトに関連するセキュリティ情報をチェックし、その情報に基づいてアクセスを付与または拒否します。たとえば、ユーザーに特定の品目への書込みアクセスがない場合、ATG Control Center ではその品目が灰色の文字で表示されます。さらに、特定のオブジェクトが特定のユーザーに表示されない場合もあります。ATG Control Center は、価格表リポジトリに含まれているすべての品目について、このセキュリティ情報をチェックできます。

詳細は、『ATG Web Commerce Programming Guide』を参照してください。

カタログへのアクセス権限は、価格表の場合と同じように設定されます。

たとえば、NDAP カタログにおける Content QA 管理者のアクセス権限を設定するには、次の操作を行います。

1. ATG Control Center の「カタログ管理」→「カタログ」タスク・セクションで、「Customer Catalogs」フォルダを選択します。
2. 「NDAP」カタログを選択して右クリックし、「アクセス権限の設定」を選択します。



3. 「アクセス権限の設定」ダイアログ・ボックスで、**「追加」**ボタンをクリックします。
4. 「ロール」タブを選択します。
5. 「グローバル・ロール」を展開し、「Control Center ユーザー」を展開して、これまでに作成されたすべての Control Center グループを表示します。
6. 「Content QA group」を選択し、「OK」をクリックします。Content QA が「アクセス権限」ダイアログ・ボックスに追加されます。

アクセス権限をリストしている列で、「リスト」ボックスおよび「読取り」ボックスを選択します。

13 販売組織としての Motorprise の操作

ATG Control Center を使用して、新規 Motorprise 顧客の作成方法を理解し、セキュリティ・アクセスの設定方法を確認できます。

この章は次の項から構成されています。

新規アカウントの作成
ATG Control Center における新規組織の設定方法、組織に契約、カタログおよび価格表を割り当てる方法、組織のロールおよびユーザーを作成する方法について説明します。

セキュリティ権限の操作
それぞれ異なるカタログおよび価格表へのセキュリティ・アクセスが適用される Motorprise ATG Control Center ユーザーの作成方法を示します。

新規アカウントの作成

新規の Motorprise 顧客アカウントを設定するには、Motorprise 従業員が最初に、ATG Control Center を使用して新規組織および会社管理者（その組織の従業員）を作成する必要があります。管理者がいることで、その管理者が新規サブ組織およびユーザーを作成できます。

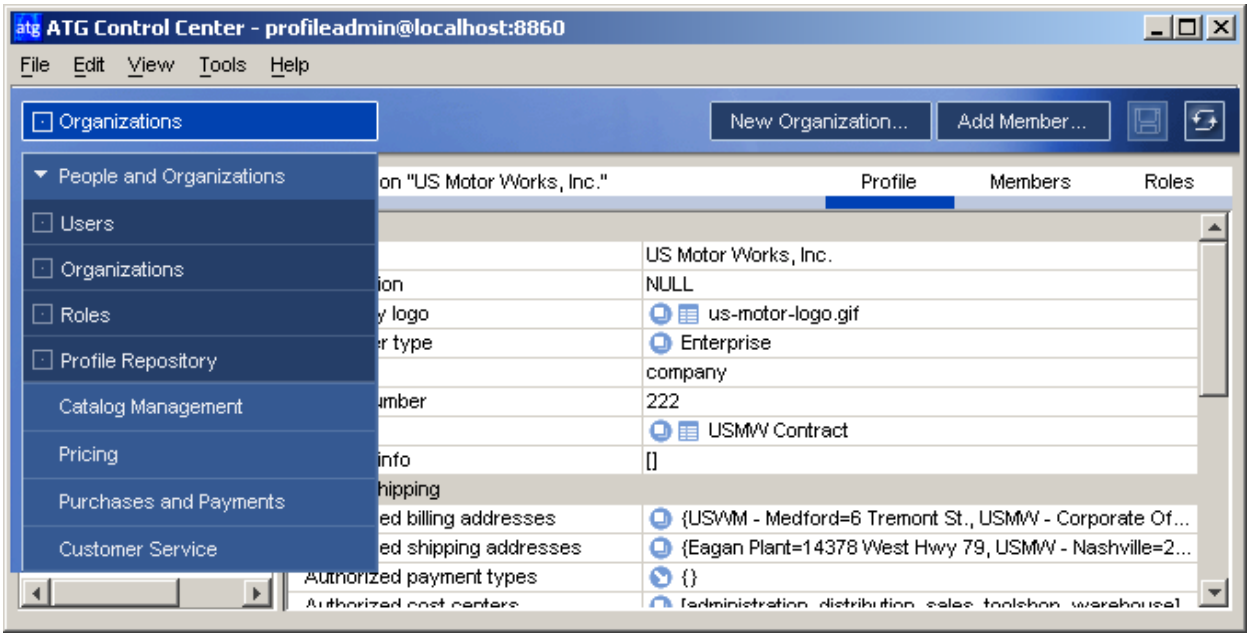
ユーザーは、所在地や支払方法など、大部分の情報を組織プロファイルから継承します。一部のプロパティは上書きできます。ただし、個人ユーザー情報は個別に入力する必要があります。この操作は、Motorprise サイトで個人または管理者が行うか、Motorprise 従業員が ATG Control Center で行います。すべての組織情報は、ATG Control Center または Motorprise サイトの「Company Admin」セクションで入力する必要があります。

新規組織の追加

Motorprise の管理者として Samantha Gresh が作成されており、Samantha は新規組織を作成できます。彼女としてログインすることで、新規組織を追加するプロセスを体験できます。

1. ユーザー名およびパスワードに profileadmin を使用して、ATG Control Center にログインします。

   Samantha は ATG Control Center の特定の領域にのみアクセスできます。たとえば、ページとコンポーネントまたはシナリオを表示および編集することできません。



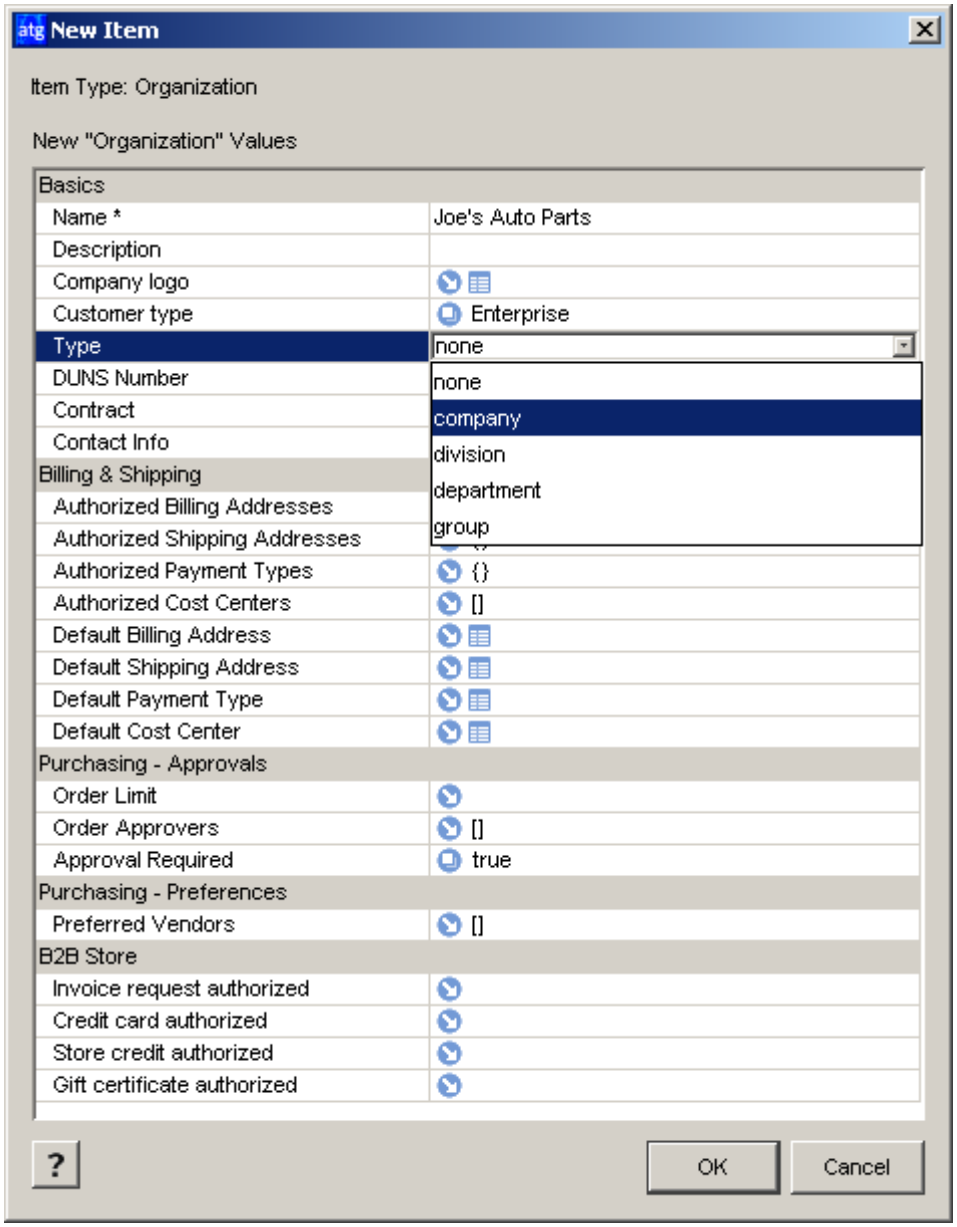
Samantha は ATG Control Center の特定の部分にのみアクセスできる。

2. 「個人および組織」→「組織」を選択します。「組織」ウィンドウが表示されます。
3. ウィンドウの左ペインで、左側の組織ツリーから「顧客組織」を選択します。

   新規組織アカウントは最上位の組織のため、「顧客組織」を選択する必要があります。これは、この後に作成されるサブ組織の親組織になります。US Motorworks, Inc.など、既存の組織を選択すると、作成する組織が USMW のサブ組織になります。
4. 画面の右上にある「新規組織」ボタンをクリックします。「新規項目」ダイアログ・ボックスが表示され、組織の属性がリストされます。
5. この組織のプロファイル・プロパティの値を指定します。アスタリスクが付いているプロパティは必須です。他のプロパティはオプションです。

   次の情報を入力します。

   - 会社名
   - 顧客タイプ（企業、優先または標準）
   - タイプ（これは新規アカウントのため、「会社」を選択します。他のオプションはサブ組織用です）



6. 「OK」をクリックします。

「会社のロゴ」フィールドの横のアイコンに注目してください。

継承値を持つ導出プロパティであることを示します。サブ組織は親組織からこのプロパティを継承します。この例の場合、作成している組織自体が親のため、このプロパティを継承することはありません。

別のリポジトリ項目を参照(リンク)するプロパティであることを示します。このフィールドは、ユーザーが入力する文字列ではなくリポジトリ項目によって入力されます。

「顧客タイプ」の横のアイコンに注目してください。

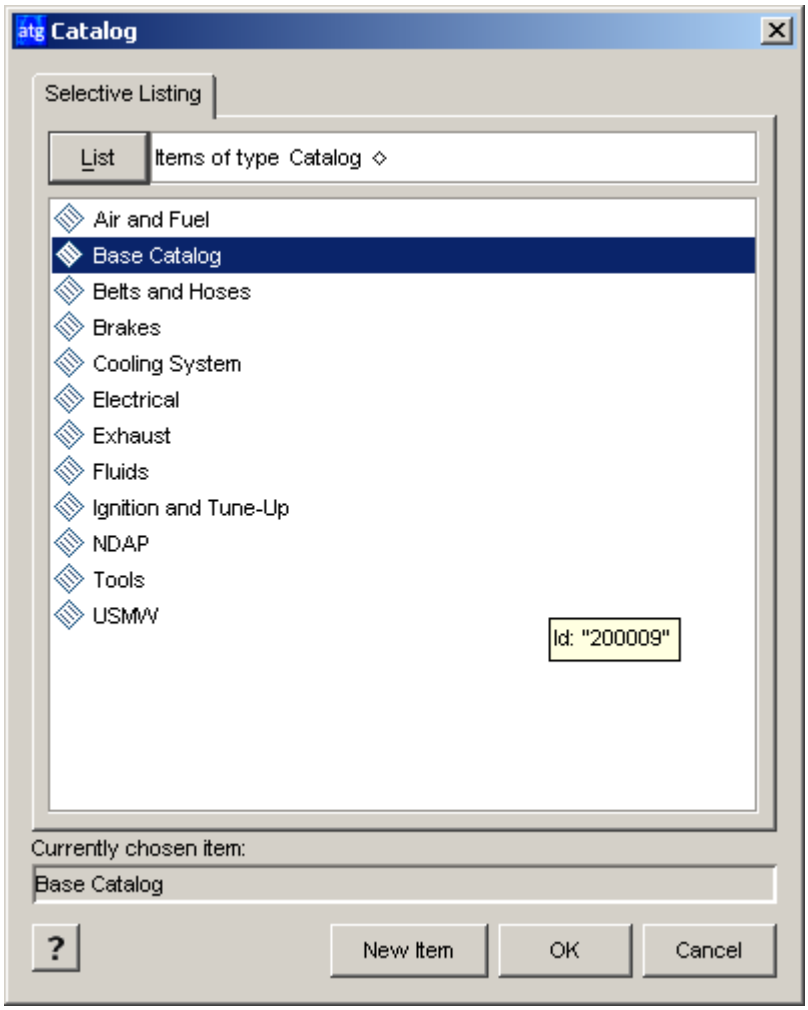
ローカルに上書きされた導出プロパティ値であることを示します。このプロパティは特にこの組織用に設定され、継承されません。

プロパティ・フィールドにアイコンがない場合は、親組織からプロパティを継承できないことを意味します。

契約の作成

契約によって顧客とカタログおよび価格表が関連付けられます。

1. 先ほど作成した組織を選択します。
2. 「契約」フィールドをハイライトし、「**...**」ボタンをクリックします。
3. 表示されるダイアログから「新規項目」をクリックします。次のプロパティを入力します。
   - 名前
   - カタログ: 「**...**」ボタンをクリックし、表示されるダイアログから「**リスト**」ボタンをクリックします。「Base Catalog」を選択するか、作成する新規カタログにサブカタログを追加することによって新規顧客カタログを作成します。詳細は、このマニュアルの「B2B パーソナライズ」を参照してください。



- 価格表: 「**...**」ボタンをクリックし、表示されるダイアログから「**リスト**」ボタンをクリックします。「Default Price List」を選択します。詳細は、このマニュアルの「B2B パーソナライズ」の「*価格表の作成*」の項を参照してください。

4. 「OK」をクリックします。

組織プロファイルに追加情報を入力できます。

所在地の入力

1. 「認可された請求先所在地」などの所在地フィールドを選択し、「**...**」ボタンをクリックします。
2. 「**追加**」ボタンをクリックし、「**新規項目**」ボタンをクリックします。
3. 所在地情報を入力し、「OK」をクリックします。
4. 新規所在地の名前を指定します。この名前は所在地テーブルのキーになります。
5. 所在地ダイアログの「OK」をクリックします。

支払タイプの入力

Motorprise で使用される支払タイプは、クレジット・カードおよび請求書です。新規組織がクレジット・カードを使用する場合は、それを入力できます。クレジット・カードを入力する場合は、組織の「承認済のクレジット・カード」フィールドを true に設定する必要もあります。会社が請求書のみを使用する場合は、クレジット・カードを入力しないでください。「承認済の請求書要求」は true、「承認済のクレジット・カード」は false に設定します。

会社がコスト・センターを使用する場合は、それを入力できます（既存のコスト・センターは他の組織に属しているため、選択しないでください。Motorprise サイトの「Company Admin」セクションには、その組織に適用されるコスト・センターのみが表示されます）。

会社が承認プロセスを使用する場合は、「要承認」フィールドを true に設定し、「オーダー制限」フィールドに金額を入力します。オーダー制限が 0 の場合、すべてのオーダーに承認が必要です。オーダー承認者は、該当のユーザーを作成し、そのロールを割り当てた後で指定します。

会社管理者の作成

次に、新規組織の会社管理者を作成します。

1. 「メンバーの追加」ボタンをクリックし、表示されるダイアログで「新規項目」をクリックします。
2. 次のフィールドを入力します。
   - ログイン名
   - パスワード
   - 名
   - 姓
   - メンバー: true に設定
   - ロケール: Motorprise は英語（en_US）およびドイツ語（de_DE）をサポートします。
   - E メール・アドレス
   - E メールの受信

   他のすべての情報は組織から継承されるか、ユーザーが後から自身で入力できます。
3. 「OK」をクリックします。

ロールの作成

新規組織のロールを作成する必要があります。

1. 「個人および組織」→「ロール」を選択します。
2. 「組織ロール」を開き、リストから自分の新規組織を選択します。
3. 右上の「新規ロール」ボタンをクリックします。

   Motorprise では、管理者、バイヤーおよび承認者に相当する 3 種類のロールを使用します。新規会社で承認が不要の場合は、この組織の最初の 2 つのみを作成する必要があります。
4. 最初に、次の情報を入力して管理者ロールを作成します。
   - 名前 – Admin
   - 摘要 – 会社管理
   - 機能 – admin（Motorprise サイト上のターゲット・コンテンツに対して使用するため、小文字で指定する必要があります）
5. バイヤー・ロールを作成します。
   - 名前 – Buyer
   - 摘要 – サイトで購買する
   - 機能 – buyer（小文字で指定する必要があります）
6. 承認者ロールを作成します。
   - 名前 – Approver
   - 摘要 – バイヤーのオーダー制限を超えるオーダーを承認する
   - 機能 – approver（小文字で指定する必要があります）

Motorprise サイトの「Company Admin」セクションで管理者がサブ組織を作成すると、これらのロールが自動的に作成されます。

ロールの割当て

作成した管理者ユーザーに Admin ロールを割り当てます。

1. 「個人および組織」→「ロール」ウィンドウで、割り当てるロールを選択します。
2. 「ファイル」→「メンバーの追加」を選択します。「メンバーの追加」ダイアログ・ボックスが表示されます。
3. 「組織」タブを選択し、ロールを割り当てる組織を選択し、「OK」をクリックします。
4. ユーザーに対して同じ操作を行います。

ユーザーが Motorprise にログインして、購入できる状態です。念のために組織のプロファイルをチェックして、少なくとも 1 つの請求先所在地と出荷先所在地があり、「請求書要求」または「承認済のクレジット・カード」が true に設定されていることを確認します。続けてバイヤーのプロファイルをチェックして、この情報を組織から継承していることを確認します。

詳細は、『ATG Web Commerce パーソナライゼーション・ガイド』の「*訪問者プロファイルの設定*」の*組織の作成*に関する項を参照してください。

セキュリティ権限の操作

価格表およびカタログにアクセスするための固有の権限を持つ、4 人の Motorprise ユーザーが作成されています。これらの Motorprise 従業員としてログインすることで、異なるレベルのセキュリティの作成方法を理解できます。

価格表およびカタログ・リポジトリを使用してセキュア・リポジトリが実際に示されるようになっています。ビジネス・ニーズに応じて、プロパティであっても、他のセキュリティ・レベルを設定できます。

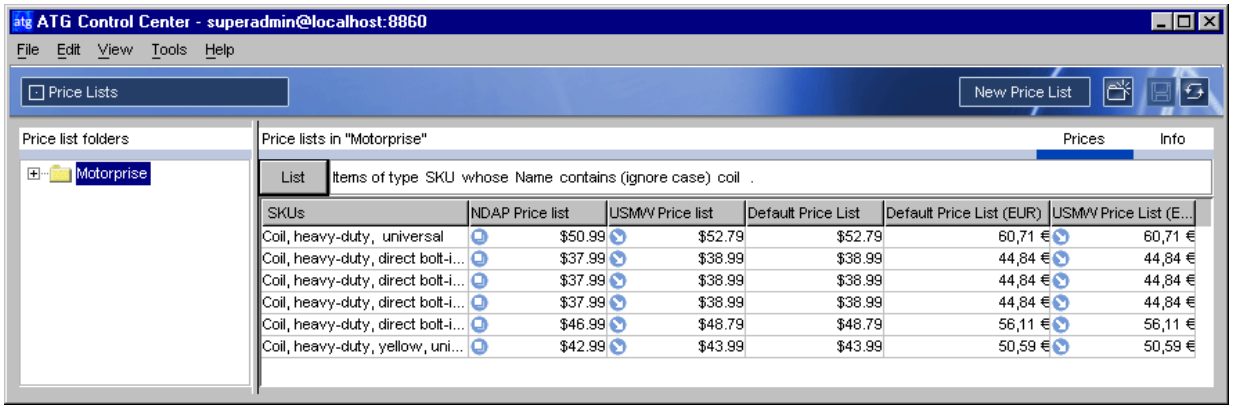
Motorprise 従業員

ユーザー	ロール	権限	ユーザー名/パスワード
Manoj Parrish	Motorprise Superadmin	すべての価格表およびカタログの読取りおよび編集が可能	superadmin
Eddie Sherfuddin	Motorprise Content QA	すべての価格表およびカタログの読取りのみ可能	contentqa
Sam Streb	Motorprise アカウント・マネージャ - NDAP	NDAP 価格表およびカタログの読取りおよび編集が可能、デフォルト価格表およびカタログの読取りのみ可能、他の顧客の価格表またはカタログの表示不可	ndapmanager
Manny Shaw	Motorprise アカウント・マネージャ - USMW	USMW 価格表およびカタログの読取りおよび編集が可能、デフォルト価格表およびカタログの読取りが可能、ヨーロッパ向けのデフォルト価格表の読取りが可能(ドイツの部署を管理するため)、他の顧客の価格表またはカタログの表示不可	usmwmanager

Manoj Parrish として(ユーザー名およびパスワードに superadmin を使用)、ATG Control Center にログインします。アクセスできるのは、「カタログ管理」、「価格設定」、「購買および支払」および「顧客サービス」セクションのみです(ユーザー名およびパスワードに admin を使用してログインすると、ATG Control Center のすべてのセクションにアクセスでき、コンポーネントやメディアなどにアクセスできます)。

「価格設定」→「価格表」セクションに進み、「Motorprise」フォルダをクリックします。Manoj は、defaultPriceList、defaultPriceList_EUR、NDAP Price list および USMW Price List にアクセスできます。

名前に Coil が含まれる SKU を検索してみましょう。組織ごとに異なる価格が表示されます。任意の価格をクリックし、変更します。Manoj は任意の価格表を編集できることがわかります。



ATG Control Center における価格表の編集

ATG Control Center をログアウトし、別のユーザーとして再びログインして、その価格表およびカタログ権限を確認します。

14 要求処理パイプライン・サーブレット

この章では、Motorprise 用に作成された 2 つの要求処理パイプライン・サーブレットについて説明します。

CheckSessionExpiration
このパイプライン・サーブレットは、ユーザーのセッションが失効しているかどうかをチェックします。

SetCurrentLocation
このパイプライン・サーブレットは、ユーザーがいる場所に応じてサイトのトップ・ナビゲーション・バーをハイライトします。

CheckSessionExpiration

セッションはユーザーの要求で作成されるオブジェクトであり、相互のやり取りが続く間、ユーザーは継続してセッションを使用できます。セッションは、複数のページ要求にわたって状態とユーザー・アイデンティティを保持するために使用されます。

ユーザーが最初にページにアクセスするときに、セッションが作成されます。以降のすべてのページ要求では、同じセッションが保持されます。

ユーザー・アクティビティがないことによるセッションの失効

事前に定義された時間にわたってユーザーのアクティビティがない場合、セッションは失効、つまり無効になります。デフォルトでは、セッション・タイムアウトは 30 分で、この値は `/atg/dynamo/servlet/session/SessionManager` の `sessionInvalidationTime` プロパティで設定されます。セッションが失効すると、ユーザーがページへのアクセスを試みても認識されず、予期したページが提供されません。たとえば、ユーザーが精算ページを要求した後、30 分以上何もしなかった場合、ユーザーが戻ってオーダーを発行しようとしても、エラーが生成されます。

Motorprise では、このようなセッション失効エラーが回避されるようになっています。そのために、`CheckSessionExpiration` サーブレットが使用されています。これはクラス `atg.projects.b2bstore.servlet.WACheckSessionExpiration` のインスタンスで、要求処理パイプライン内で起動されます。これがユーザーのセッションをチェックします。セッションが有効な場合は、目的のページに要求を渡します。セッションが失効している場合は、エラーを生成するのではなく、ユーザーをログイン・ページにリダイレクトし、再びログインすることを指示します。このサーブレットの `relativeExpirationURL` プロパティが、ユーザーのリダイレクト先のページを指定します。

登録ユーザーも匿名ユーザーも、セッション失効について同じ方法が使用されています。

フェイルオーバーによるセッションの失効

セッションを処理しているサーバーが異常終了し、別のサーバーがそのセッションに対処する場合、要求のセッション ID は無効になります。新規セッションが作成されて、旧セッションのすべての状態および ID 情報を復元します。DynamoHttpServletRequest の sessionRestored プロパティは、旧セッションの ID に設定されます。そのため、Motorprise 内でセッション ID が無効な場合は sessionRestored をチェックして、このセッション・フェイルオーバー条件の結果でないことを確認します。

<ATG11dir>Motorprise/config/atg/projects/b2bstore/servlet/にある CheckSessionExpiration のプロパティ・ファイルで、insertAfterServlet プロパティが /atg/dynamo/servlet/pipeline/SessionSaverServlet に設定されます。

これは、<ATG11dir>MotorpriseJSP/j2ee-apps/motorprise/config/atg/projects/b2bstore/servlet/CheckSessionExpiration にある CheckSessionExpiration プロパティ・ファイルです。

```
$class=atg.projects.b2bstore.servlet.WACheckSessionExpiration

# Specify where in the servlet pipeline this servlet should appear
insertAfterServlet=/atg/dynamo/servlet/pipeline/SessionSaverServlet

# Specify the web application registry where Motorprise is registered
webAppRegistry=/atg/registry/webappregistry/ServletContextWebAppRegistry

# Specify the name under which the Motorprise web app is registered
webApplicationName=MotorpriseJSP

# Specify the root path (relative to the web app context root) of URLs'
# that we should check for session expiration

relativeExpirationPath=/

# Specify the URL (relative to the web app context root) that we should
# redirect to if an expired session is detected.

relativeExpirationURL=/
```

CheckSessionExpiration の relativeExpirationPath プロパティおよび relativeExpirationURL プロパティを構成して、Web アプリケーションのコンテキスト・ルートを基準とする相対 URL を使用し、これらを実行時に絶対 URL に変換します。

SetCurrentLocation

Motorprise には、「Company Admin」、「Product Catalog」、「My Account」、「Current Order」など、様々なページ・セクションがあります。各セクションは様々なページで構成されています。ユーザーがこのようなセクションを参照しているとき、トップ・ナビゲーション・バーでそのタイトルがハイライトされます。

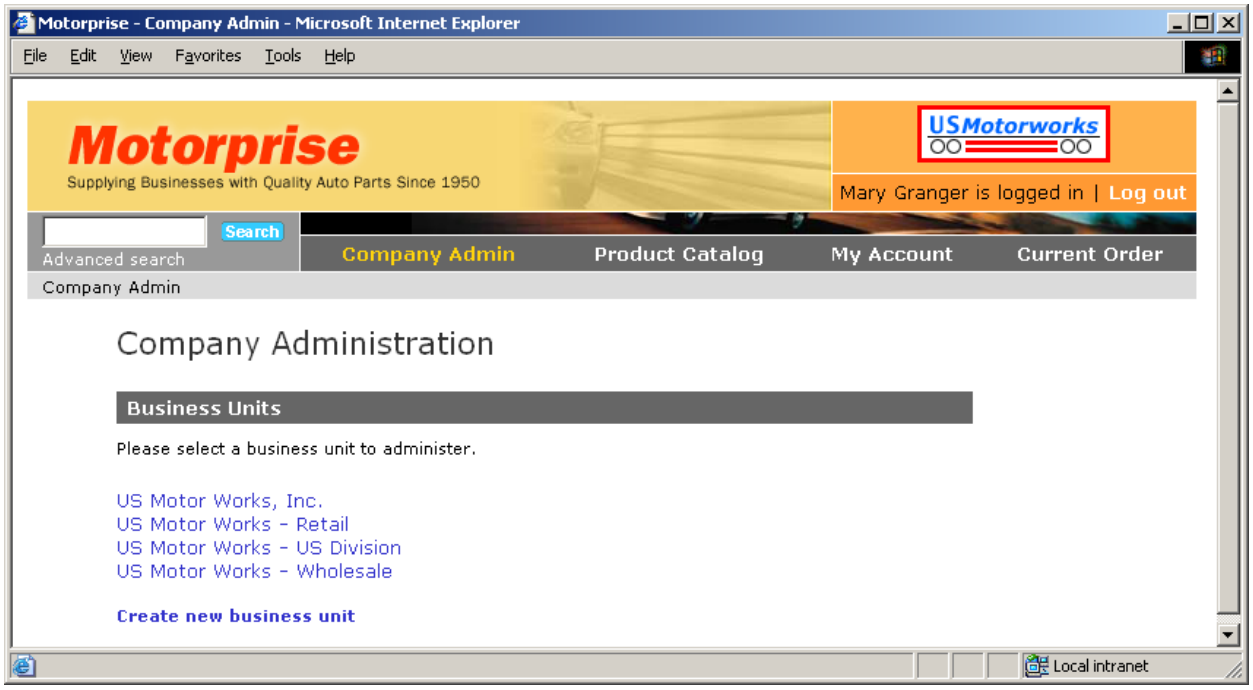
ユーザーの場所を特定し、対応するセクション・タイトルをハイライトするために、SetCurrentLocation,というサーブレットが開発されました。これはクラス

atg.projects.b2bstore.servlet.WASetcurrentLocation のインスタンスで、要求処理パイプラインの中で起動されます。

ユーザー用の一時プロパティ currentLocation が新しく作成され、ユーザーがアクセスしているセクションがここに格納されます。/en/common/BrandNav.jsp からの次のスニペットは、このプロパティをチェックして該当のセクションをハイライトする方法を示しています。

```
<dsp:droplet name="Switch">
  <dsp:param bean="Profile.currentLocation" name="value"/>
  <dsp:oparam name="admin">
    <td align="center"><dsp:a href="../admin/business_units.jsp">
    <b><font color="#FDD30E" size=-1>Company Admin</font></b></dsp:a></td>
  </dsp:oparam>
  <dsp:oparam name="default">
    <td align="center"><dsp:a href="../admin/business_units.jsp">
    <b><font color="#FFFFFF" size=-1>Company Admin</font></b></dsp:a></td>
  </dsp:oparam>
</dsp:droplet>
```

場所が admin に等しい場合は、次の画面ショットに示されているように、Motorprise の「Company Admin」セクションが黄色でハイライトされます。



ユーザーが「Company Admin」セクションにいる場合は、そのタイトルが黄色でハイライトされる。

SetCurrentLocation のマップ・プロパティ relativePathMap は、Web アプリケーションのコンテキスト・ルートを基準とする相対 URL を使用して、ストアの URL と仮想部分の間のマッピングを指定します。これらは実行時に絶対 URL に変換されます。このプロパティには、各セクション（「Company Admin」、「Product Catalog」、「My Account」および「Current Order」）の相対ドキュメント・パスと、ユーザーがそのいずれかのパ

スからページにアクセスする場合に location プロパティに設定される値が含まれます。これは、MotorpriseJSP/j2ee-apps/Motorprise/config/atg/projects/b2bstore/servlet/SetCurrentLocation のプロパティ・ファイルです。

```
$class=atg.projects.b2bstore.servlet.WASetCurrentLocation

# Specify where in the servlet pipeline this servlet should appear
insertAfterServlet=/atg/userprofiling/ProfileRequestServlet

# Specify the web application registry where Motorprise is registered
webAppRegistry=/atg/registry/webappregistry/ServletContextWebAppRegistry

# Specify the name under which the Motorprise web app is registered
webApplicationName=MotorpriseJSP

# Set URL's relative to the Motorprise context root.

relativePathMap=/en/catalog/=product_catalog,\
            /en/admin/=admin,\
            /en/home.jsp=product_catalog,\
            /en/checkout/=current_order,\
            /en/user/=my_account,\
            /de/catalog/=product_catalog,\
            /de/admin/=admin,\
            /de/home.jsp=product_catalog,\
            /de/checkout/=current_order,\
            /de/user/=my_account,\
            /ja/catalog/=product_catalog,\
            /ja/admin/=admin,\
            /ja/home.jsp=product_catalog,\
            /ja/checkout/=current_order,\
            /ja/user/=my_account
```

ユーザーが新規ページを要求すると、URL が relativePathMap プロパティ内のパスと比較されます。サーブレットは、マップのキーであるドキュメント・パスを昇順でソートし、要求内のディレクトリ・パスの有無をチェックします。完全一致がある場合は、対応する値が location プロパティに割り当てられます。完全一致がない場合は、URL が属する最も近いディレクトリが検索され、そのディレクトリの値が location に追加されます。

サーブレットの開始

SetCurrentLocation および CheckSessionExpiration をサーブレット・パイプラインに挿入するためには、これらを Oracle Commerce Core Platform と同時に開始する必要があります。これを実現するために、/atg/dynamo/servlet/Initial コンポーネントに次の行が追加されています。

```
initialServices+=/atg/projects/b2bstore/servlet/SetCurrentLocation,\
            /atg/projects/b2bstore/servlet/CheckSessionExpiration
```

これらのサーブレットは、<ATG11dir>/Motorprise/src/Java/atg/projects/b2bstore/servlet/に置かれています。

索引

X

あ

い

お

か

く

け

こ

さ

し

す

せ

た

つ

て

な

は

ひ

ふ

ほ

ま

ゆ

よ

り