Visual Modeler

ベスト・プラクティス・ガイド

リリース 9.1

Visual Modeler

ベスト・プラクティス・ガイド

リリース 9.1

お願い

本書および本書で紹介する製品をご使用になる前に、 43 ページの『特記事項』に記載されている情報をお読みください。

目次

第 1 章 バックアップおよびリカバリーのベスト・プラクティス

最善のリカバリー計画では、予防を重視します。堅固な環境のセットアップ、冗長システムの配備、定期的なバックアップおよびリストアのポリシーの確立、およびバックアップからリカバリーできることを確認するための定期的なテストの実施により、実装しているインフラストラクチャーの各層を保護して、災害の影響を限定的にすることができます。

バックアップおよびリストアのポリシーに関する決定は、ビジネスの基準に基づいて行う必要があるものがあります。毎日サイトが利用できるような状態にすることはビジネスにとってどのような価値があるか。どの程度の量のデータであったら失っても大丈夫か。e-commerce の Web サイトを顧客が利用できなくなっても許容できるのはどのぐらいの期間か。各種のバックアップおよびリカバリー・ソリューションにはどのような時間およびコストのトレードオフがあるか。これらの質問に答えることは、バックアップおよびリカバリーの要件を決定する際に役立ちます。

最も単純なバックアップ・システムとしては、テープまたはその他のリモート・デバイスにデータとアプリケーションのコピーを保存し、それを遠隔地にあるデータ・センターに保管するという方法が考えられます。それよりも優れた戦略は、実装環境の各部分に冗長システムを配備して、1 つのシステムに障害が発生しても、もう 1 つのシステムが利用できるようになっているというものです。サイトのミラー・イメージを遠隔地に保持し、そのイメージを定期的に実動環境のデータと同期させるのが、最も堅固なソリューションです。後者のソリューションは、高価になりますが、即座のリカバリーが可能です。前者のソリューションは、後者よりは安価ですが、後者よりもリカバリーを実行するために時間と労力がかかります。

デプロイメント・アーキテクチャーの概要

充実した開発環境を整えると、サイトの更新や保守が容易になるだけでなく、リカバリーの一環として完全なアプリケーションの再ビルドが必要になる場合も迅速に行えます。デプロイメント・アーキテクチャーは、以下の要素から構成されます。

- ビルド環境: 以下のものを含む、デプロイメントをビルドするために必要なすべての要素が用意されている、予測可能な既知のビルド環境
  - JDK's
  - SDK's
  - コード・リポジトリー (CVS など)
  - ライブラリー

  コーディングから実動に至るまでに必要なステップはさまざまであり、反復が必要になる場合もありますが、ビルド環境には、予測可能な方法でデプロイメントを再ビルドする必要がある場合に必要となるすべてのものが含まれている必要があります。
- QA エリア: 品質保証タスクを実行するための独立した環境。QA は、(おそらく) 複数のエンジニアが作成したものを 1 つの単位として実行できるように統合する最初の環境です。

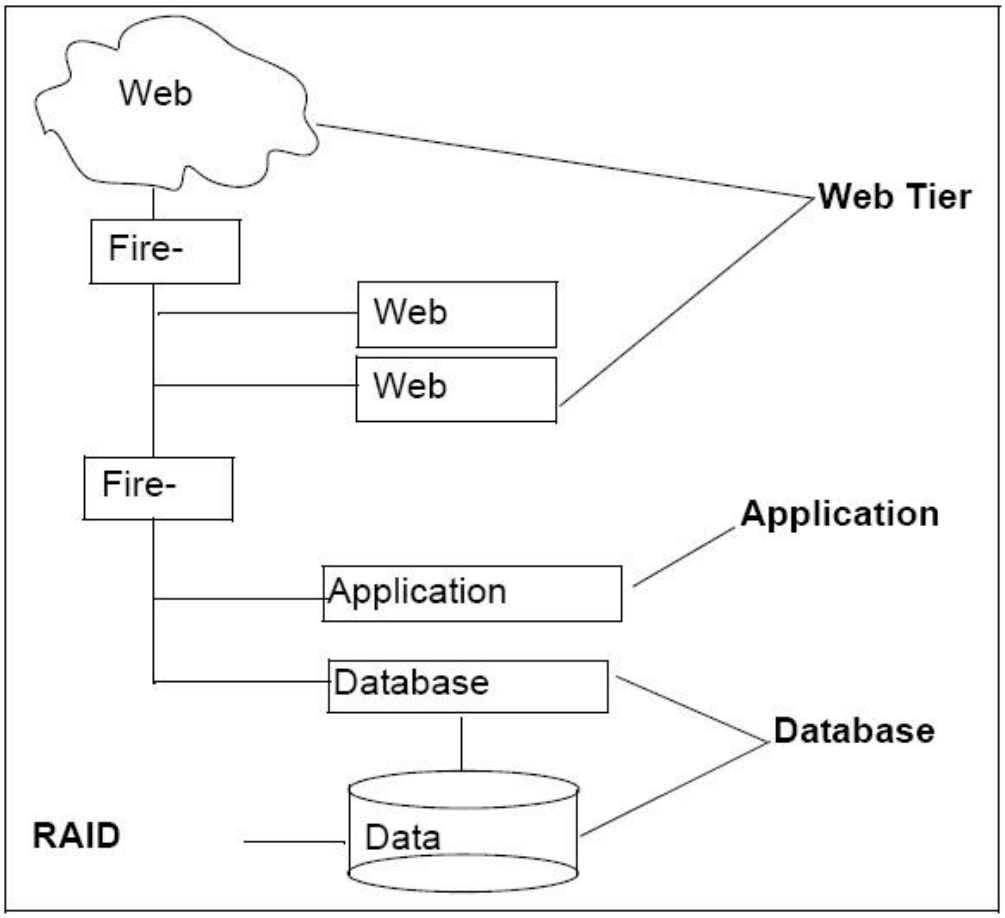
- ステージング・エリア: QA で統合されたものが、今度は本番環境をエミュレートする具体的なコンテキストで実行されるようになる独立した環境。

デプロイメント・インフラストラクチャー

堅固なインフラストラクチャーを確立するための一般的な戦略は、「すべてのものを二重にする」ことです。つまり、冗長システムを配備して、1 次システムに障害が発生したときに、できる限り迅速に 2 次システムをオンラインにできるようにすることです。以下の図は、3 つの層からなる典型的なインフラストラクチャーを示しています。

- Web 層: Web ブラウザーからの要求を処理し、Web ブラウザーにコンテンツを提供するすべてのコンポーネント。
- アプリケーション層: Web 層からの要求を処理し、通常、データベース層からのデータで、Web 層に動的なコンテンツを提供するすべてのコンポーネント。
- データベース層: アプリケーション層にデータを提供するすべてのコンポーネント。

以下の図は、典型的なデプロイメント・インフラストラクチャーの図です。



2 つの Web サーバーによって、1 つに障害が発生しても、もう 1 つが稼働し続けることが保証されます。 第 2 のファイアウォールを設置してデータをさらに保護すると、データ保護に関する特定の法規制要件も満たすことができます。

データを物理的に保護する戦略の 1 つとして、RAID デバイスに実動データを保管する方法があります。1 つのドライブに障害が発生しても、データは失われません。このような戦略には機械的なしきい値が存在します。構成にもよりますが、しきい値を超える数のドライブに障害が発生すると、データが失われます。これは要件を決めるときに考慮しなければならないことです。

バックアップ戦略

データの保護やアプリケーション・サーバーおよび Web サーバーのリカバリーのためのバックアップ戦略にはさまざまなものがあります。ほとんどのバックアップ戦略は、アプリケーション層および Web 層ですでに実行されているもののコピーを保存するというものになります。

データベース・リカバリー戦略

バックアップ戦略によって、災害発生後にどのぐらい速くデータをリカバリーできるかが決まります。データベースを再稼働させるまでに許容できるタイムライン (OS の再ビルドやデータベースの再ロードが必要な場合はその時間も含む) を決定し、それに応じてバックアップ・ポリシーを計画します。

実行する必要があるデータベース・バックアップを以下に示します。

- チェックポイント・バックアップ: トランザクションがコミットされると、トランザクション・レベルでデータベース・サーバーがアクティビティーをログに記録します。チェックポイント・バックアップは、最後のチェックポイント・バックアップの実行以降のトランザクションをログに書き込みます。チェックポイント・ログを別の物理デバイスに書き込みます。これにより、トランザクション・レベルでデータベース・アクティビティーのスナップショットが作成されます。データベースに障害が発生しても、アクティビティーを復元するための一連のレコードが存在します。
- 実装環境でチェックポイント・バックアップを実行する間隔は、ビジネスの状況に応じて決定します。サイトで毎時間数百万ドルの取引が行われている場合は、1 時間の間に数回チェックポイント・バックアップを行うことを推奨します。サイトで毎時間発生するトランザクションの数がわずかである場合は、チェックポイント・バックアップを実行する回数を減らすことができます。ビジネスの活動のレベルに応じたデータ・リカバリーを可能にする間隔を決定します。
- 日次増分バックアップ: 増分バックアップでは、各日に変更されたファイルのみを保存します。日次増分バックアップを、物理メディアを使用するのではなく、別の施設に保存するのがよい戦略です。このバックアップは、実際上、ディスクからディスクへのコピーです。
- 週次フルバックアップ: フルバックアップでは、最後のバックアップ以降に変更されたファイルだけでなく、データベース全体を保存します。フルバックアップを別の施設に保存するのがよい戦略です。

データベースのリトリーブ

このタスクについて

通常は、以下のステップを実行して、データベースをリカバリーします。

手順

1. 最後のバックアップ・データから初期のデータベースをリストアします。
2. 日次増分バックアップから日付順にデータをリストアします。
3. チェックポイント・バックアップを使用して、最後の数時間のアクティビティーを復元します。

アプリケーション・サーバーと Web サーバーのリカバリー

アプリケーション・サーバーや Web サーバーのリカバリー・ポリシーを計画する場合、アプリケーション・サーバーまたは Web サーバーのいずれかに障害が発生した場合にそれ以前とまったく同じ状態の代替サーバーを構築できるように計画します。これは「代替サーバーを構築して継続」という原則です。

アプリケーション・サーバーおよび Web サーバーの固有のデプロイメントを構成しているすべてのもの (構成ファイル、プロパティー・ファイル、JVM、CVS リポジトリーからの元のソース・コードなど) のコピーを必ず用意しておきます。カスタム JSP ページなど、Web 層に維持されているすべての静的データをバックアップします。バックアップ・プロセスには、ソース・コードとは見なされないためにソース・コード・リポジトリーに保持されていない場合がある、Web サーバーおよび Web コンテナーの構成ファイルやサイトの運用に必要なその他のファイルも確実に対象として含めるようにします。

サーバーを再構築し、運用可能な状態に戻すための開始点として QA 環境およびステージング環境を使用します。

第 2 章 ユーザー・インターフェースのアイコンの理解

以下の表は、Visual Modeler で使用するアイコンについて説明しています。

表 1. アイコン表

アイコン	説明
	「**モデルの編集 (Edit Model)**」 - このアイコンは、モデルを編集するのに使用します。

第 3 章 アイテム・モデリング

アイテム・モデリングのための絶対パスおよび相対パスの使用

このトピックでは、プロパティーやルールなどのエンティティーを指定するために絶対パスまたは相対パスを使用する方法を取り上げます。パスの形式は、以下のとおりです。

<モデル・グループのルート・ノード>.<プロパティーまたはルールがある、オプション・アイテムへのパス>.<プロパティー名またはルール名>

例として、プロパティー memoryProvided への次の絶対パスを挙げます。

MXDS-7500.memory.sim256.memoryProvided

モデル・グループのルート・ノードは MXDS-7500 です。プロパティーが memoryProvided のオプション・アイテムへのパスは memory.sim256 です。memoryProvided はプロパティー名です。

複数のモデルでプロパティーまたはルールを使用する予定である場合は、特殊記号を使用して相対パスを指定できます。例えば、次のパスでは "*" は、パスがモデル・グループ階層のルートから始まることを示しています。

*.memory.sim256.memoryProvided

ピリオド (.) で始まるパスは、「ルールのアタッチ・ポイントから」であることを示します。例えば、次のパスの "." は、「現在のモデル内の memory と呼ばれるオプション・クラス内の sim256 というオプション・アイテム」を示します。

.memory.sim256

アイテム・モデリングの計画

モデルは、構成可能なアイテムを表します。IBM Sterling Web の実装を計画する際には、アイテムのモデルの設計方法を検討することから始めます。特定のアイテムのモデルを作成するための唯一の「正しい」方法は存在しません。一方、技術的に正しくても、非効率で維持が難しいモデルが作成される方法が無限にあります。

念頭に置いておくべきトレードオフを以下に示します。

- コスト要因: 作成、維持、およびパフォーマンス。

  モデルの維持コスト、モデルに期待されるパフォーマンスと比較して、モデルの作成にかかるコストを調整する必要があります。モデルの作成コストは、その開発に費やした 1 回限りの労力を表し、モデルの維持コストは、モデルを維持し拡張するためにある期間にわたって費やした労力を表します。一方、パフォーマンスは、特定のハードウェア・プラットフォーム上でのモデルの実行速度を表します。 これらのいずれの要素も最適化できますが、複数の要素の最適化を試みようとすると、ほとんどの場合、競合する目標が生じることに留意してください。例

えば、複雑なモデルの実行速度は高速かもしれませんが、維持が難しくなります。モデルの概要およびベスト・プラクティスの設計原則については、『アイテム・モデリングの設計上の考慮事項』 を参照してください。

- 実装者の役割: モデルの作成のみ担当、モデルの作成と維持を担当、モデルの維持のみ担当。

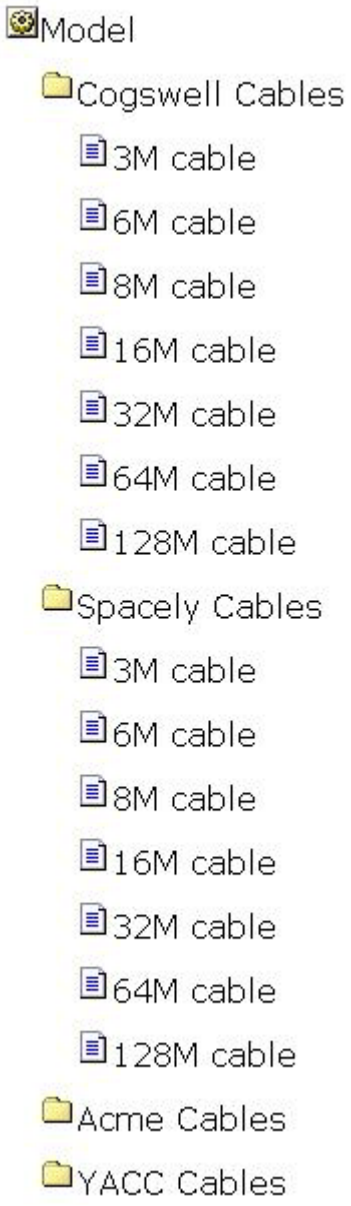
  実装者の役割は、実装者が何を重視するかにも影響します。例えば、あるコンサルタントがモデルの実装に 1 カ月与えられ、そのモデルの維持は別のグループに託されるとします。この場合、コンサルタントは維持が容易なモデルを設計することよりも迅速にモデルを設計することを重視する可能性があります。実装者がモデルの維持も担当する場合は、モデルの設計により多くの時間をかけ、モデルの実行速度はさほど高速でなくなる可能性もありますが、長期的には維持しやすいモデルが作成されるはずです。実装者の役割が何であろうと、目標は将来問題が発生することを回避するモデルを作成することです。

アイテム・モデリングの設計上の考慮事項

アイテム・モデルを設計する際には、複雑なモデルを効率的に実行可能にする設計を目指すという原則を実践することが重要です。

アイテム・モデルのサイジング

モデルのサイズは重要です。モデルが大きくなるほどブラウザーでのレンダリングに時間がかかり、維持が難しくなり、構成時にコンフィギュレーターは価格の取得、ルールの起動などのためにモデル構造を何度も「行き来」することになります。このトピックで説明するサブアセンブリーやその他の手法によりモデルのサイズをできる限り小さく保つと、パフォーマンスが向上し維持コストが低減されます。例えば、複数のサプライヤーからケーブルを購入していて、各サプライヤーが長さが異なるケーブルをいくつも提供しているとします。ケーブルの数量、長さ、およびサプライヤーをユーザーが選択できるようにしたいとします。そのための方法の 1 つとして、以下の図に示すように、個々の利用可能なオプションすべてを表すオプション・アイテムをモデルに作成する方法があります。

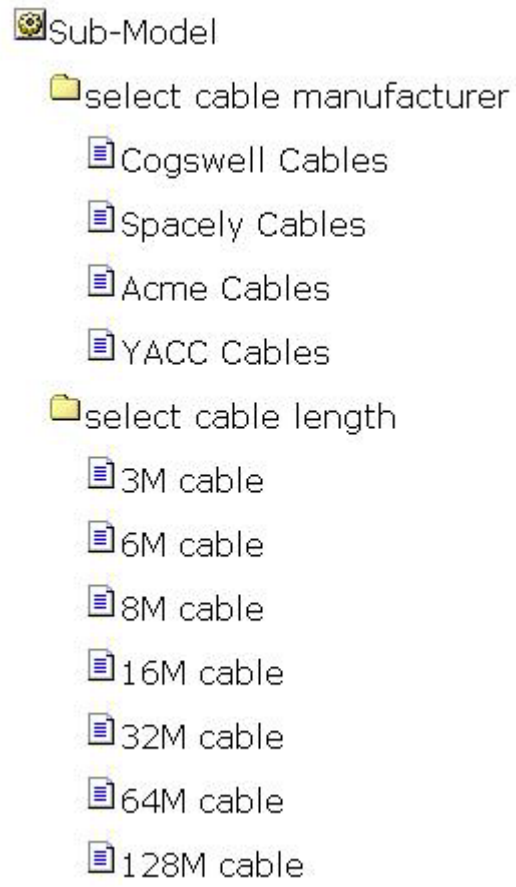


このアプローチでもうまくいきますが、エンド・ユーザーが関心を示す可能性がまずない多くのオプションを含むモデルを作成、実装、および維持するのは徒労となるかもしれません。

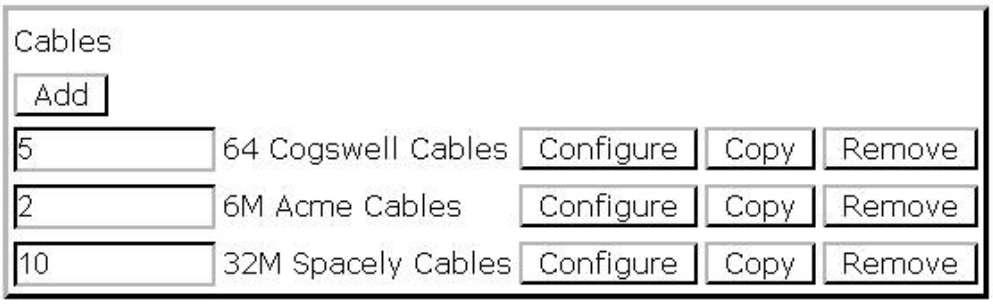
この代わりとなる方法として、オプション・アイテム・グループとして異なるケーブル長のオプション・アイテムを実装して、各メーカーの下にケーブル長のオプション・アイテム・グループを含めるという方法があります。この方法では維持が容易になります。モデル作成者は、1 箇所でケーブルのオプション・アイテム情報を更新できます。ただし、このアプローチでは、エンド・ユーザーはケーブルの非常に大きなリストから選択しなければならなくなり、行き来しなければならない大きなモデルが依然として存在するためパフォーマンスも向上しません。

これよりもよい方法として、サブモデルを作成する方法があります。以下の図のようなサブモデルを作成することで、ユーザーは、ケーブルのメーカーと長さを選択してから、動的なインスタンス生成により、必要に応じてさまざまなタイプや長さ

のケーブルを追加できるようになります。



以下の図は、動的なインスタンス生成により、エンド・ユーザーがケーブルの選択肢を構成可能にする、ケーブル選択 UI の例を示しています。



明らかに、このアプローチはモデルを小さく保ちます。モデル作成者が大量の重複したオプション・アイテムに対応する必要がなくなるため、このモデルの維持は容易になります。モデルが小さいため、パフォーマンスが向上します。また、適切なものを見つけるためにケーブルのタイプやメーカーの長いリストを検索する必要がなくなるためエンド・ユーザーにとって構成がらくになります。

ポップアップ・コントロールによるアイテムの数量の指定

モデル作成者は、ユーザーがアイテムを選択してから、必要な数量を入力できるようにしたい場合があります。これを実現するための最良の方法は、Option Class

Display に popup-qty を設定するという方法です。エンド・ユーザーがアイテムを選択すると、数量ボックスが表示され、アイテムの必要な数量を入力できます。

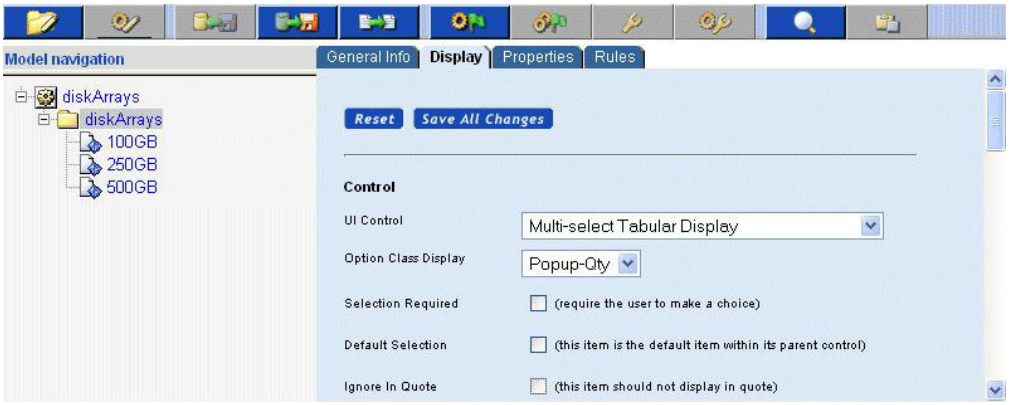
数量ボックスを配置することを好まないモデル作成者もいます。数量ボックスを配置する代わりに、User Entered Values (UEV) コントロールを使用して、アイテムの横に編集フィールドを表示し、そこにユーザーが数量を入力できるようにします。問題は、popup-qty コントロールの動作が、UEV コントロールの動作とかなり異なるという点です。popup-qty コントロールには数量処理が組み込まれていますが、UEV コントロールには追加作業が必要になります。

エンド・ユーザーが popup-qty ボックスに数量を入力すると、アプリケーションは自動的に選択されたアイテムの数量を選択します。アイテムにアタッチされたプロパティーは、コンフィギュレーター状態 (プロパティー・プール) に含まれ、数値プロパティーの値は入力された値で乗算されます。

UEV コントロールで値が入力されると、それ以外のことは何も起こりません。UEV コントロールは、ユーザーからのいくつかの追加情報を単に取り込むように設計されています。UEV が数量を処理するようにするには、モデル作成者は、UEV コントロールに入力された値に基づいて、その数だけ選択されたアイテムを選択する拡張ルールを作成する必要があります。UEV コントロールに入力された値を使用して割り当てルールにより _quantity プロパティーを設定する方法は期待どおりにうまくいきません。なぜなら、この方法ではプロパティー・プールにアイテムのプロパティーのインスタンスが自動的に作成されないからです。

選択されたアイテムの横に popup-qty ボックスを表示するには、popup-qty Option Class Display スタイルと数量コントロールを備えたテーブル表示の 1 つを使用します。このようにすると、正しい数のアイテムが確実に選択され、適切なプロパティーがプロパティー・プールにコピーされます。

例えば、次の図は、Visual Modeler を使用して popup-qty コントロールをセットアップする方法を示しています。「モデルおよびグループ (Models and Groups)」パネルから、変更したいモデルを選択して、「**モデルの編集 (Edit Model)**」アイコンをクリックします。「Model navigation」ページが表示されます。以下の図に示すように、変更したいオプション・グループをクリックし、「**Display**」タブをクリックしてから、「**UI Control**」ドロップダウン・リストから「Multi-select Tabular Display」を選択します。



ページの一番下までスクロールし、「Column Headings」、「Column Properties」、および「Column Alignment」の設定値を入力します。次の図は、設定値の例を示しています。

最後に、モデルをコンパイルしてテストします。「Product Configurator」ページで指定した配置で popup-qty コントロールが表示されるはずです。

モデルのプロパティーの定義

プロパティーは、IBM Sterling Configurator ではいたるところで使用されます。モデル作成者は、プロパティーをモデル、オプション・クラス、およびオプション・アイテムにアタッチし、メッセージの表示、アイテムの表示、非表示、または選択に加えて、その他のプロパティーの値さえも設定するためにそれらのプロパティーに作用するルールを作成します。プロパティーの定義方法や使用方法を決定する際には、構成可能なアイテムのモデルを作成する上でプロパティーが果たす重要な役割を考慮して、ある程度慎重に行う必要があります。このセクションでは、プロパティーを定義してアタッチする際に役立ついくつかのヒントや従うべき手順の概要を示します。

モデル・プロパティー名の定義

モデルを開発するとき、特に時間の制約が厳しい場合、モデル作成者は開発プロセスをスピードアップするために簡単な方法をとろうとすることがよくあります。よく使われる方法の 1 つに、プロパティーをしばしば曖昧あるいはわかりにくい短い名前で作成する方法があります。この方法は、短期的にはモデルの開発をスピードアップするかもしれませんが、モデルの維持に必要な労力は大幅に増えます。モデル作成者は、常に特定のプロパティーが何を表すのかすぐに明らかになるようにモデルを設計する必要があります。プロパティーに付ける名前が有意義であればあるほど、現在および今後のモデルのデバッグおよび維持が容易になります。

以下の例を検討します。

model node	properties
Model	tmr=sum(mr) tmp=sum(mp) amr=value(tmr)-value(tmp)
OC1	
oi1	mr=1
oi2	mr=2
OC2	
oi3	mp=2
oi4	mp=1

一見しただけでは、このモデルに割り当てられたプロパティーが何を達成しようとしているのか明らかになりません。有意義な名前を作成するようにもう少し時間をかけると、すべてのプロパティーの本質や各プロパティーの相互の関係を理解するのがかなり容易になります。

model node	properties
Model	total_mem_required=sum(mem_required) total_mem_provided=sum(mem_provided) additonal_mem_required=value(total_mem_required) - value(total_mem_provided)
OC1	
oi1	mem_required=1
oi2	mem_required=2
OC2	
oi3	mem_provided=2
oi4	mem_provided=1

固有のモデル・プロパティーの定義

しばしば、特定の機能の実装を急ぐあまりに、モデル作成者は、目前の問題のために特別設計した新たなプロパティーを作成する代わりに、既存のプロパティーを再使用します。これには以下の 2 つの影響が考えられます。

- 既存のプロパティーの名前が目前の問題と無関係である場合は、モデルを理解するのが難しくなる可能性がある。
- 再使用がプロパティーの元の用途と矛盾する場合は、プロパティー名の再使用はモデルで実際にエラーの原因になる可能性がある。

前のセクションの例をもう一度検討してみます。モデル作成者が必要なメモリーを保存するためのプロパティーを作成し、そのプロパティーに “memory”という名前を付けていたとします (この名前がなぜそもそもよくない選択であったかについては、前のセクションを参照してください)。今度は、提供されたメモリーを保存するためのプロパティーが必要であると判断し、memory という名前のプロパティーがあることに気付き、新たなプロパティーを作成する代わりにそのプロパティーを使用することを決定したとします。

そうなると、一見して、2 つのアイテムがメモリーを必要とし、2 つがメモリーを提供する代わりに、すべてのオプション・アイテムにある程度の量のメモリーが必要であるかのように見えます。それだけではなく、total_memory_required プロパティーは、必要なメモリーと提供されたメモリーの両方の合計の計算を実行するようになったため、その値は正しいものでなくなります。このような形にモデルを作成すると、モデル作成者は、必要なプロパティーの特定のインスタンスを分離するために、適切なプロパティー・インスタンスが含まれるアイテムへの絶対パスまたは相対パスを使用するなど、余分な作業を行わなければならなくなります。

モデル・プロパティーの階層の定義

プロパティーは、ルート・モデル・グループ・レベルから個々のモデル・レベルに至る、モデル・グループ階層のあらゆるレベルで定義できます。プロパティーをどこに定義するかによって、どのモデルがそのプロパティーを見ることができ、利用できるかが決まります。モデルの設計時に少し考慮すれば、モデルの開発が迅速になり、プロパティーが複雑になるのを回避するのに役立ちます。以下のガイドラインに基づいて、プロパティーを定義する場所を決定します。

プロパティーを特定のモデルのみで使用する場合は、プロパティーをそのモデルのレベルでのみ定義します。

プロパティーを特定のモデル・グループ内の複数のモデルで使用する場合は、そのモデル・グループのレベルでプロパティーを定義します。

プロパティーを複数のモデル・グループにまたがるモデルで使用する場合は、そのプロパティーを使用するすべてのモデル・グループを含む最初のモデル・グループでプロパティーを定義します。

最後の手段として、プロパティーをルート・モデル・グループ・ノードで定義します。

同じ値を持つ複数のプロパティーの定義

同じ値を持つ複数のプロパティーを定義すると、モデルの作成や維持が容易になる場合があります。この考え方は最初はわかりづらいと思われるため、最善の方法として例を挙げて説明します。ディスク・アレイの選択肢からユーザーが選択することを可能にするモデルを作成するとします。ディスク・アレイのそれぞれのタイプには、いくつかのディスクが関連付けられているとします。ユーザーは任意のタイプの複数のディスク・アレイを選択できます。モデル作成者が計算する必要のある情報の 1 つに、ユーザーが選択したディスクの総数があります (以下参照)。

model node	properties
DiskArray SubModel	total_disks=sum(disks)
100GB disk arrays	
blue	disks=4
mauve	disks=6
250GB disk arrays	
blue	disks=8
mauve	disks=12

今度は、100GB ディスク・アレイの数と 250GB ディスク・アレイの数も把握する必要があることに気付いたとします。必要なプロパティーへのアイテム・パスを指定してこれらの値を計算したり、モデル内の特定のところにアタッチする必要のあるルールを作成したり、またはすべての disk および total_disk プロパティーを手直しする代わりに、古いディスク・プロパティーと同じ値を持つ新しいプロパティーをいくつか定義するだけですみます (以下参照)。

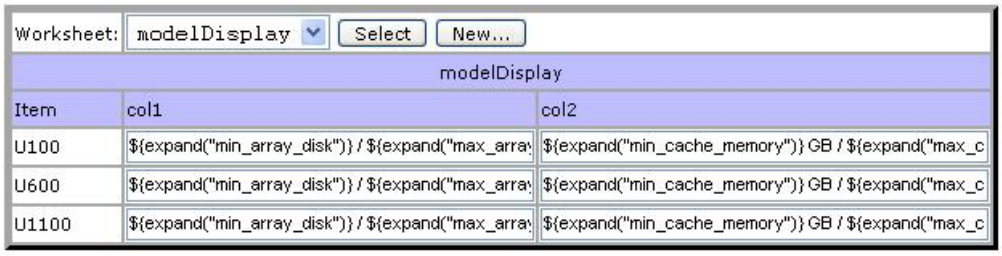
model node	properties
DiskArray SubModel	total_disks=sum(disks) total_100GBdisks = sum(100GB_disks) total_250GBdisks = sum(250GB_disks)
100GB disk arrays	
blue	disks=4 100GB_disks=4
mauve	disks=6 100GB_disks=6
250GB disk arrays	
blue	disks=8 250GB_disks=8
mauve	disks=12 250GB_disks=12

次に、ディスクの総数を求めたい場合は、依然として sum(disks) で求めることができます。個々の値が必要な場合は、それらの値も得ることができます。すべて、個々のプロパティーへのパスを指定することも、すでに完了している処理を変更することもなく、総数を求めることができます。

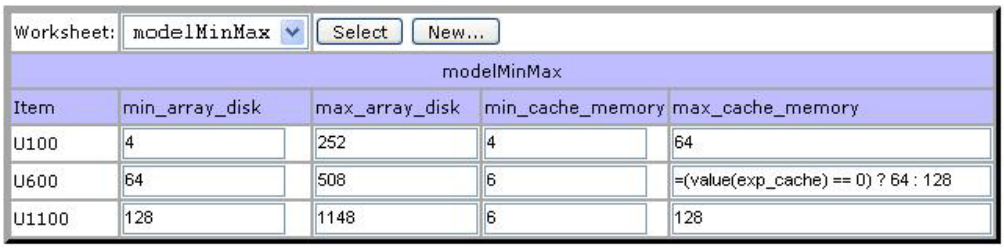
ワークシートを使用したプロパティーの割り当て

モデルを開発する際には、複数のオプション・クラスまたはオプション・アイテムにプロパティーの同じセットを割り当てる必要がある場合がよくあります。このような場合にワークシートが非常に役立ちます。その理由は、ワークシートが任意の数のオプション・クラスまたはアイテムの特定のプロパティーに値を迅速に設定することを可能にするからです。これは、複数の場所にプロパティーの値を設定する式を使用する場合に特に役立ちます。モデル作成者は、必要なすべてのアイテムに式をコピーして貼り付けるだけですみます。この例を以下に示します。この例で

は、テーブル表示内の各アイテムに設定された一部の表示プロパティーが示されています。ワークシートを使用して、col1 と col2 の式をオプション・クラスの各アイテムに簡単に切り貼りできるようにします。



ワークシートを使用すると、モデルの特定の部分の概略がわかるというさらなるメリットもあります。少し考慮して計画すると、ワークシートでモデルの特定の部分の概要を示したり、特定の問題に対する解決策全体を示すこともできます。以下に同じオプション・クラスを異なる基準で表示したものを示します。この場合は、各オプション・アイテムに設定されたすべての min および max プロパティーを確認することが目的です。



プロパティーの式でのチェーニングの回避

アイテムにアタッチされたプロパティーには、順序という概念はありません。つまり、プロパティーの値を設定するために式を使用する場合、どの特定の順序で行われる式の評価にも依存できないということです。プロパティー A に式が含まれ、プロパティー B にプロパティー A に依存する式が含まれる場合、式 B から作成されたルールが、式 A から作成されたルールの後に起動される保証はありません。この問題を回避するには、モデル作成者には次の 2 つの選択肢があります。

- 最初の式を 2 番目の式が評価される前に起動されるルールにします。式から生成されるすべてのルールの優先順位は 50 です。 最初の式のルールを作成し、その優先順位を 50 未満に設定することにより、プロパティー A の値がプロパティー B の値が計算される前に設定されるようにすることができます。
- 繰り返しルール起動をオンにします。この場合、ルール起動の最初のフェーズでプロパティー A の値が計算されます。ルール起動ループの 2 番目のフェーズで、最初のパスで計算されたプロパティー A の値に基づいてプロパティー B の値が計算されます。注: このような方法で多数の式をチェーニングすると、すべての条件を満たすために通らなければならないルール起動ループのパスの数により、パフォーマンスが低下する可能性があります。このため、最初の選択肢を推奨し、式のチェーニングをできる限り回避することをお勧めします。

ルールの定義

ルールの条件の定義

ルールの条件は、関係式にブール演算を適用することにより作成されます。関係式は、より小さい、等しい、より大きい、IN、NOT IN などの関係演算子を使用して、 1 つの関数/プロパティーを別の関数/プロパティーと比較します。結果は、真または偽となります。AND や OR などのブール演算子は、これらの関係式のセットを包含します。関係式は、ルールの断片であるため、フラグメント と呼ばれます。関係演算子の左側は LHS と省略される場合が多く、RHS は右側を意味します。

ルールのフラグメントの順序付け

ルールのフラグメントの評価によっていつルールが起動されるかが決まるため、ルール内のフラグメントの順序は重要です。ルールが真または偽のいずれであるかを迅速に判断できるほど、モデルの効率も向上します。また当然、ルールを起動すべきでないとモデルが迅速に判断できるほど、モデルが別の処理に迅速に進むことができます。ルールの起動を回避する可能性が最も高いものから最も低いものの順にルールのフラグメントを配置すると、パフォーマンスを向上することができます。

ルールが適切なときにだけ起動されるように、必ずルールをテストします。どのような状況でルールの結果が使用されたり使用されなかったりするかを把握するのも重要です。例えば、常に起動されるが、式の結果がゼロになる場合、または拡張セクションの > および <= フィールドに式に一致するものがない場合に、拡張セクションにある何かを選出しない拡張ルールは、非常に非効率です。

汎用ルールの作成

可能な限り、できる限り汎用のルールを作成します。例えば、次のルールは、productType および handsetType プロパティーがアタッチされたいずれのアイテムにもアタッチできます。

```
If  propval(productType) != value(selectProductType)
and propval(handsetType) != value(phonePreference)
    set _isVisible=0
```

このルールは、productType プロパティーがアタッチされ、かつ選択された製品タイプに一致せず、かつ選択された電話の好みが現在のアイテムの好みに一致しないアイテムの場合にのみ起動されます。このような汎用ルールは、次のような多くの具体的なルールを置き換えることができます。

```
If  propval(productType) == literal("handset")
and propval(handsetType) != literal("camera")
and value(phonePreference) == literal("camera")
    set _isVisible=0
If  propval(productType) == literal("handset")
and propval(handsetType) != literal("flip")
and value(phonePreference) == literal("flip")
    set _isVisible=0
...
```

式を使用したルールの定義

多くの場合、ルールの代わりに式を使用することができます。モデル作成時には、式は、実行時に評価される式を値として持つアタッチされたプロパティーとして維持されます。式で参照されるいずれかの関数が評価できない場合は、式は起動されていないルールのように機能します。マルチパス・ルールの起動がオンになると、式は、ルールの起動が終了するまで、または式が結果を生成するまで、各起動パスで再評価されます。

結果を計算することを求める唯一の条件が、式で使用される関数/プロパティーが値を持つという条件のときは、ルールではなく式を使用します。

例えば、ユーザーが選択したトラックのコンポーネントが適切であるように、アクセルやホイールベースなどのトラックのコンポーネントの回転半径を計算したいとします。次のように turningRadius を計算するために、式を該当するトラックのコンポーネントにアタッチできます。

```
turningRadius = value(axelTurnFactor) * value(wheelBaseTurnFactor)*
sum(turningElements)
```

この式は、value(axelTurnFactor)、value(wheelBaseTurnFactor)、および sum(turningElements) の各式がすべて数値の結果を生成すると起動されます。

これに相当するルールは、次のようになります。

```
if (value(axelTurnFactor) >= 0 or value(axelTurnFactor) < 0)
and (value(wheelBaseTurnFactor)>= 0 or value(wheelBaseTurnFactor)< 0)
and (sum(turningElements) >=0 or sum(turningElements) <0)
      turningRadius = value(axelTurnFactor) * value(wheelBaseTurnFactor)*
sum(turningElements)
```

ルールの条件の部分は、かなり長く、常に真に評価されるように見えます。ただし、関数が参照するプロパティーが存在しない場合は関数は NULL を戻すことがあるため、このルールは実際には戻り値が >= 0 または < 0 であるかどうかを評価して結果が NULL でないことをチェックしています。

ルールでのパス情報の回避

ルール・フラグメントの LHS および RHS は、関数とプロパティー名から構成されます。プロパティー名には、相対パス情報と絶対パス情報の両方を含めることができます。ルール・フラグメントにプロパティーのパス情報を指定すると、パス情報またはオプション・クラスが変更された場合、ルールが実行できなくなる場合があります。

例えば、次のルールは、完全指定のパス情報を使用して wheelSize および wellSize を参照しています。モデル作成者がホイールまたはフェンダーのいずれかのオプション・クラスの名前を変更する必要がいずれある場合、または他のモデルでルールを再利用したい場合に、ルールが正しく実行されない可能性があります。

```
If value(*.wheels.wheelSize) == literal("17in")
and value(*.fender.wellSize) < literal(17)
    set _isVisible=0
```

プロパティーの 1 つの特定のインスタンスにアクセスしたい場合に限り、かつこの値を保持するための新しいプロパティー・タイプを作成できない場合にのみパス情

報を使用します。プロパティーのパス名を参照しなければならない場合は、絶対パス名ではなく相対パス名を使用するほうがよい場合が多くあります。

制約テーブルとルールの比較

このセクションでは、顧客の選択肢を制限するために制約テーブルを使用する場合とルールを使用する場合のトレードオフについて説明します。制約テーブルは、あるオプション・アイテムに対して顧客が選択するものに基づいて、顧客が選択する 1 つ以上の別のオプション・アイテムを制限します。例えば、自動車の外装色として選択した色が、内装色の選択を制限するなどです。

制約テーブルは、単純な検証に最も適しています。例えば、あるオプション・アイテムが別のオプション・アイテムとともに問題なく使用できるかどうかなどの検証に適しています。単純な制約テーブルは、ルールよりも維持が容易です。しかし、大きい複雑な制約テーブルは、維持が難しく、パフォーマンスの問題につながる場合があります。

制約テーブルは、内部ではルールに変換されます。

ルールは、複雑な検証問題を表現するのに最適であり、制約テーブルよりも汎用性があります。制約テーブルもルールもエラー・メッセージを表示できますが、プロパティーを設定したり選択を行うルールを作成することもできます。

モデルのパフォーマンスの向上

このセクションでは、モデルの作成および維持を簡略化するためのいくつかの手法について説明します。適切な手法を選択すると、モデルのパフォーマンスにかなりの効果があります。

- オプション・クラス・グループ、オプション・アイテム・グループ、およびサブアセンブリーの使用:

  この手法は、オプションのグループが多数の異なるモデルで繰り返し使用される場合にうまく機能します。

  例えば、販売するすべてのコンピューターにユーザーが選択できるハード・ディスクのリストが含まれているとします。オプション・クラス・グループ、オプション・アイテム・グループ、およびサブアセンブリーを作成すると、モデル作成者は 1 カ所で共通の情報を作成して維持し、その情報を多くのところで使用できます。

  この方法の 1 つの欠点は、同じモデルにサブアセンブリーが何回も含まれる場合、モデルが非常に大きくなる可能性があるということです。

- サブモデルのパンチ・インとパンチ・アウト:

  この手法は、構成に、構成可能な選択が含まれる場合に役立ちます。サブモデルのパンチ・インおよびパンチ・アウトを使用して、1 つの販売モデル全体の中に複雑な構成をネストできます。

  1 つの欠点は、構成されたアイテムのすべてのコピーの構成が同じになることです。

- 動的なインスタンス生成:

  この手法は、単一モデル内の 1 つの構成済みアイテムの複数インスタンスを可能にします。各インスタンスは、異なる構成にすることができます。

モデリング・ツール

モデルの作成には時間がかかり長い単調な作業になる場合がありますが、最終的にはモデルの正確さと作成したソリューションの拡張容易性がプロジェクトの成功の鍵となります。拡張が容易で正確なモデルの作成を支援する目的で、開発のさまざまなフェーズでモデル作成者を導くツールの集合を開発しました。開発時には、モデル作成者がどのモデルをデバッグすべきか判断するのにトレース・ログとモデル・レポート・ツールが役に立ちます。モデルを実動に移す前に、ロード・テスト・プラットフォームを使用してモデルの拡張容易性と安定度をテストできます。最後に、実行時には、モデルのシステムの使用状況に関する洞察をモデル・キャッシュ状況ページから得ることができ、メガバイトに相当する価値のログ情報を解釈するためにログ・アナライザーを使用できます。

トレース・ログの使用

トレース・ログは、ルール・エンジンの実行状況を示します。これは、常にではないですがしばしば、コンフィギュレーターがサーバーに対して行う各要求の最も時間のかかる部分となります。トレース・ログの目的は、正しく機能しないルールのデバッグに必要な情報の提供とルールの実行時間の追跡であるため、必ずトレース・ログをレビューすることからデバッグを始めてください。

トレース・ログは Visual Modeler を使用して作成します。そのためには、以下の手順を実行します。

1. 「Model Group」ナビゲーションに進み、デバッグしたいモデルにナビゲートします。
2. 「Models and Groups」パネルからモデルを選択します。
3. モデルが「Model Preview」タブに表示されます。
4. 「Test」アイコンをクリックします。
5. モデルは別のウィンドウで実行されます。
6. 「Debug」をクリックします。

トレース・ログが別のウィンドウに表示されます。

ログは、2 つのセクションから構成されています。最初のセクションは、ルール起動トレースです。2 番目のセクションは、ルールの起動の終了時の状態のプロパティー・プールです。

以下の図は、ルール起動トレースのセクションの例を示しています。

Rule Firing Trace

#	(ms)	Result
0	0	Applying picks
1	0	Firing phase [0]:begin
2	0	Firing rules on MXDS-7500.Disk Drives
3	0	MSG_E_Available_HDD_Slots ==> fires on TRUE - priority = 50
4	0	Property not found [MX75_HDD_Ordered or MX75_Bays_Available], taking null action
5	0	**took 0ms.**
6	0	Firing rules on MXDS-7500.Memory
7	0	MSG_E_MX75_Memory_Software_Check ==> fires on TRUE - priority = 50
8	0	TESTING:sum(MX75_Mem_Ordered) <sum(MX75_Mem_Required) [nullreturn=false]
9	0	FALSE: 0.0 < 0.0
10	0	FALSE: sum(MX75_Mem_Ordered) <sum(MX75_Mem_Required) [nullreturn=false]
11	0	**took 0ms.**
12	0	Firing rules on MXDS-7500.Placeholder for auto memory selection
13	0	ASG make placeholder invisible ==> fires on TRUE - priority = 50
14	0	Left side property [MX75_Mem_Auto_Select] not found, taking null action
15	0	**took 0ms.**
16	0	Firing rules on MXDS-7500.AutoMemory
17	0	EXP_MX75_Automatic_Memory_Selection ==> fires on TRUE - priority = 50
18	0	Left side property [MX75_Mem_Auto_Select] not found, taking null action
19	0	**took 0ms.**
20	0	Firing rules on MXDS-7500.Software.Application
21	15	EXP_MX75_Fire_Wire ==> fires on TRUE - priority = 50
22	15	Left side property [MX75_Video_Editing] not found, taking null action
23	15	**took 15ms.**

ルール起動トレースには、以下の 3 つのカラムがあります。

- シーケンス番号。これは、ルールの問題について他の人に伝える場合に役立ちます。「Xxx となっている 42 行目が見えますか?」と伝えるのは簡単です。
- 経過時間。ログが記録された時からルールの起動が開始されるまでにかかった時間を記録します。
- トレース・ログの本文。これは、評価される条件、行われている割り当て、ルールの開始またはルールの終わりなどのルールの起動のさまざまな側面を示します。

ログは、各ルール起動項目の次にルールの起動に必要なミリ秒数を示します。モデルの実行に要したミリ秒数の合計が、ルール起動トレースの最後に記録されます。

プロパティー・プール・トレースにも 3 つのカラムがあります。

- Name は、アイテムおよびそのアイテムのプロパティーへの絶対パスの名前です。
- Type は、Numeric、List、または String などの名前付きプロパティーのプロパティー・タイプです。
- Value は、ルールが起動された後のプロパティーの値です。

以下の図は、プロパティー・プール・トレースのセクションの例を示しています。

Property Pool

Name	Type	Value
MXDS-7500.CONFIG: FIRST FIRE	Numeric	1.0
MXDS-7500.MX75_Bays_Available	Numeric	2.0
MXDS-7500.MX75_Card_Slot_Available	Numeric	4.0
MXDS-7500.MX75_Mem_Ordered	Numeric	0.0
MXDS-7500.MX75_Mem_Required	Numeric	0.0
MXDS-7500.Service Options.View Service.UI: COLUMN SPAN	Numeric	2.0
MXDS-7500.Placeholder for auto memory selection.UI: CONFIG CELL HTML CLASS	String	configsubcell_plain
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: CONFIG CELL HTML CLASS	String	configsubcell_plain
MXDS-7500.Microprocessor.UI: CONSTANT GUIDING TEXT	String	Dual processor capable motherboard, supporting Intel processors
MXDS-7500.Microprocessor.UI: CONTROL	String	RADIO
MXDS-7500.Disk Drives.UI: CONTROL	String	RADIO
MXDS-7500.Placeholder for auto memory selection.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Software.UI: CONTROL	String	CHECKBOX
MXDS-7500.Accessory Cards Message.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Accessory.Graphic Cards.UI: CONTROL	String	RADIO
MXDS-7500.Accessory.Cards.UI: CONTROL	String	CHECKBOX
MXDS-7500.Accessory.Network Cards.UI: CONTROL	String	RADIO
MXDS-7500.Service Options.View Service.UI: CONTROL	String	CHECKBOX
MXDS-7500.Placeholder for auto memory selection.UI: DEFAULT SELECTION	String	no
MXDS-7500.Accessory Cards Message.UI: DEFAULT SELECTION	String	no
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: DEFAULT SELECTION	String	no

"single user" というこのログを使用して、クリック 1 回につき、ルールがどの程度広範囲に及ぶかを調べてみます。ルールを起動するのにどのぐらい時間がかかっているか確認します。その答えが 100 から 200ms の範囲を超える場合は、拡張容易性の問題がある可能性があります。そうである場合は、トレース・ログを使用しして、適切に実行されていない特定のルールがあるか確認します。

モデル・レポート・ツールの使用

モデル・レポート・ツールは、モデルの他のモデルを基準とした相対的なサイズの概要を提供します。どのモデルをテストするか決定する際にこのツールを使用すると役に立ちます。一定期間のテストの結果を追跡できるため、モデルに対する変更の量を確認できます。

ロード・テスト・ツールの使用

ロード・テスト・ツールは、モデルをデプロイした後にそのモデルのパフォーマンスがどのようになるか判断するのに役立ちます。ロード・テスト・ツールを使用する前に行わなければならないことを以下に挙げます。

- 何をテストするのか理解します。

- 影響が何を意味するのかわかるようにテスト・ケースを分けます (ローカルとリモートの LAN のテスト、クラスタリングありとなしでのテスト、Web フロントありとなしなど)。
- モデルを変更する場合は、テストの実行やテスト・シナリオのやり直しのためのスクリプトも変更する必要があります。
- 現在のモデル・グループよりもグローバルである場合は、プロパティーを使用したいモデルの上位にあるモデル・グループ・ツリー内の最下位の位置に定義します。

Rule Firing Trace

#	(ms)	Result
0	0	Applying picks
1	0	Firing phase [0]:begin
2	0	Firing rules on MXDS-7500.Disk Drives
3	0	MSG_E_Available_HDD_Slots ==> fires on TRUE - priority = 50
4	0	Property not found [MX75_HDD_Ordered or MX75_Bays_Available], taking null action
5	0	**took 0ms.**
6	0	Firing rules on MXDS-7500.Memory
7	0	MSG_E_MX75_Memory_Software_Check ==> fires on TRUE - priority = 50
8	0	TESTING:sum(MX75_Mem_Ordered) <sum(MX75_Mem_Required) [nullreturn=false]
9	0	FALSE: 0.0 < 0.0
10	0	FALSE: sum(MX75_Mem_Ordered) <sum(MX75_Mem_Required) [nullreturn=false]
11	0	**took 0ms.**
12	0	Firing rules on MXDS-7500.Placeholder for auto memory selection
13	0	ASG make placeholder invisible ==> fires on TRUE - priority = 50
14	0	Left side property [MX75_Mem_Auto_Select] not found, taking null action
15	0	**took 0ms.**
16	0	Firing rules on MXDS-7500.AutoMemory
17	0	EXP_MX75_Automatic_Memory_Selection ==> fires on TRUE - priority = 50
18	0	Left side property [MX75_Mem_Auto_Select] not found, taking null action
19	0	**took 0ms.**
20	0	Firing rules on MXDS-7500.Software.Application
21	15	EXP_MX75_Fire_Wire ==> fires on TRUE - priority = 50
22	15	Left side property [MX75_Video_Editing] not found, taking null action
23	15	**took 15ms.**

ルール起動トレースには、以下の 3 つのカラムがあります。

- シーケンス番号。これは、ルールの問題について他の人に伝える場合に役立ちます。「Xxx となっている 42 行目が見えますか?」と伝えるのは簡単です。
- 経過時間。ログが記録された時からルールの起動が開始されるまでにかかった時間を記録します。
- トレース・ログの本文。これは、評価される条件、行われている割り当て、ルールの開始またはルールの終わりなどのルールの起動のさまざまな側面を示します。

ログは、各ルール起動項目の次にルールの起動に必要なミリ秒数を示します。モデルの実行に要したミリ秒数の合計が、ルール起動トレースの最後に記録されます。

プロパティー・プール・トレースにも 3 つのカラムがあります。

- Name は、アイテムおよびそのアイテムのプロパティーへの絶対パスの名前です。
- Type は、Numeric、List、または String などの名前付きプロパティーのプロパティー・タイプです。

- Value は、ルールが起動された後のプロパティーの値です。

以下の図は、プロパティー・プール・トレースのセクションの例を示しています。

Property Pool

Name	Type	Value
MXDS-7500.CONFIG: FIRST FIRE	Numeric	1.0
MXDS-7500.MX75_Bays_Available	Numeric	2.0
MXDS-7500.MX75_Card_Slot_Available	Numeric	4.0
MXDS-7500.MX75_Mem_Ordered	Numeric	0.0
MXDS-7500.MX75_Mem_Required	Numeric	0.0
MXDS-7500.Service Options.View Service.UI: COLUMN SPAN	Numeric	2.0
MXDS-7500.Placeholder for auto memory selection.UI: CONFIG CELL HTML CLASS	String	configsubcell_plain
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: CONFIG CELL HTML CLASS	String	configsubcell_plain
MXDS-7500.Microprocessor.UI: CONSTANT GUIDING TEXT	String	Dual processor capable motherboard, supporting Intel processors
MXDS-7500.Microprocessor.UI: CONTROL	String	RADIO
MXDS-7500.Disk Drives.UI: CONTROL	String	RADIO
MXDS-7500.Placeholder for auto memory selection.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Software.UI: CONTROL	String	CHECKBOX
MXDS-7500.Accessory Cards Message.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: CONTROL	String	controls/allpicked.jsp
MXDS-7500.Accessory.Graphic Cards.UI: CONTROL	String	RADIO
MXDS-7500.Accessory.Cards.UI: CONTROL	String	CHECKBOX
MXDS-7500.Accessory.Network Cards.UI: CONTROL	String	RADIO
MXDS-7500.Service Options.View Service.UI: CONTROL	String	CHECKBOX
MXDS-7500.Placeholder for auto memory selection.UI: DEFAULT SELECTION	String	no
MXDS-7500.Accessory Cards Message.UI: DEFAULT SELECTION	String	no
MXDS-7500.Accessory Cards Message.Accessory Card Image.UI: DEFAULT SELECTION	String	no

"single user" というこのログを使用して、クリック 1 回につき、ルールがどの程度広範囲に及ぶかを調べてみます。ルールを起動するのにどのぐらい時間がかかっているか確認します。その答えが 100 から 200ms の範囲を超える場合は、拡張容易性の問題がある可能性があります。そうである場合は、トレース・ログを使用しして、適切に実行されていない特定のルールがあるか確認します。

キャッシュ状況ツール

cmd=configstatus を使用して、キャッシュの現在の内容を表示できます。

パフォーマンス

モデルのパフォーマンスのルール

- アイテムへの過剰なパス。
- 以下の式によりメモリーを追加するルール:

```
totalMem = value(*.adapter.1.memory) + value(*.adapter.2.memory) +
value(*.adapter.3.memory) + value(*.adapter.4.memory)
```

- 以下の式よりも実行速度が低速:

  ```
  totalMem = sum(memory)
  ```

- メモリーのプロパティーが他の用途のために別の場所に存在し、その結果 sum(memory) が誤った値を生成する場合は、adapterMemory と呼ばれるアダプター・アイテムの 1 から 4 にさらにプロパティーを追加し、以下の式を使用します。

  ```
  totalMem = sum(adapterMemory).
  ```

- これは以下の式を維持するよりも維持のための労力が少なくてすみます。

  ```
  totalMem = value(*.adapter.1.memory) + value(*.adapter.2.memory) +
  value(*.adapter.3.memory) + value(*.adapter.4.memory)
  ```

- 必要な場合にのみ起動されるルールを作成します。

  totalMem = sum(mem) を割り当てるルールは、count(mem) > 0 の場合にのみ起動する必要があります。

モデルのパフォーマンスのプロパティー

- モデル・グループ階層の正しい位置にプロパティーを定義します。
  - プロパティーがこのモデル限定である場合は、このモデル内に定義します。
  - このモデル・グループ内の他のモデルが使用することがある場合は、現在のモデル・グループに定義します。
  - 現在のモデル・グループよりもグローバルである場合は、プロパティーを使用したいモデルの上位にあるモデル・グループ・ツリー内の最下位の位置に定義します。

第 4 章 データのアーカイブ

データを管理することは、ビジネスを保護する上で重要になります。実動データを定期的にアーカイブし、データベースのアーカイブを保管するための 1 次保管場所および 2 次保管場所 (可能であればオフサイトに) を設けます。日次増分バックアップ、週次フルバックアップ (業務のボリュームに応じてさらに頻繁に実施) を含む、アーカイブ活動の定期的スケジュールを定めます。

第 5 章 統計情報の更新

データベースの統計情報を更新すると、データベース照会最適化プログラムがデータベース・インデックスを再調査して、データを検索するための最も効率的なパスを再計算することができます。このセクションでは、2 つのスクリプト、1 つは Oracle データベースの統計情報を更新するためのもの、もう 1 つは SQL Server データベースの統計情報を更新するためのものを紹介します。

テーブルの適切に更新された統計情報を入手する方法については DBA に問い合わせてください。また、さらなる支援が必要な場合は、ご使用のデータベースの資料を参照してください。

Oracle データベースの統計情報の更新

以下の例は、スキーマ・レベルで Oracle データベースの統計情報を更新する方法を示しています。*schema name*、*owner name*、および*table name* を適切なスキーマ、所有者、およびテーブルの名前で置き換えます。

```
EXEC DBMS_STATS.GATHER_SCHEMA_STATS(
ownname=> 'schema name' ,
cascade=> TRUE,
estimate_percent=> DBMS_STATS.AUTO_SAMPLE_SIZE,
degree=> DBMS_STATUS.AUTO_DEGREE,
granularity=>'AUTO',
method_opt=> 'FOR ALL COLUMNS SIZE AUTO');
```

以下の例は、テーブル・レベルで Oracle データベースの統計情報を更新する方法を示しています。

```
exec dbms_stats.gather_table_stats(
ownname=> 'owner name',
tabname=> 'table name',
estimate_percent=> DBMS_STATS.AUTO_SAMPLE_SIZE,
cascade=> DBMS_STATS.AUTO_CASCADE,
degree=> null,
no_invalidate=> DBMS_STATS.AUTO_INVALIDATE,
granularity=> 'AUTO',
method_opt=> 'FOR ALL COLUMNS SIZE AUTO');
```

要件に応じて、統計情報をデータベースまたはインデックスのレベルでも更新できます。

SQL Server データベースの統計情報の更新

以下の例は、テーブル・レベルで SQL Server データベースの統計情報を更新する方法を示しています。*table name* および *index name* を該当するテーブル名とインデックス名で置き換えます。

```
UPDATE STATISTICS ON <table name> [ . <index name> ]
    WITH FULLSCAN {, NORECOMPUTE }
```

第 6 章 JVM メモリーとチューニングのガイドライン

メモリーに関連する問題に直面した場合、JVM メモリーの設定値を調整すると正常に作動する環境に戻すことができます。このセクションでは、JVM メモリーの設定値およびパフォーマンスのチューニングについてのガイドラインを示します。このガイドラインを適用するには、使用している JVM およびサーブレット・コンテナー環境に精通している必要があります。

JVM メモリーの設定値の調整

通常は、アプリケーション・サーバーを稼働している JVM にできる限り多くのメモリーを割り振る必要があります。JVM メモリーの構成を以下のように設定してこれを実現できます。

- -Xmx は、マシンの物理メモリーの 80% から 100% の値に設定する必要があります。-Xmx の値が小さすぎると、アプリケーション・サーバーに OutOfMemory エラーの障害が発生する可能性があります。-Xmx の値が大きすぎる場合は、メモリーのフットプリントが大きくなり、Java のヒープがスワップアウトされるリスクがあり、別のパフォーマンスの問題が生じます。
- -Xms には、-Xmx の設定値の約半分の値を設定する必要があります。アプリケーション・サーバーの継続的なメモリー使用量がどの程度であるか示す履歴データがある場合は、-Xms にその値に近い値を設定します。

  もう 1 つの方法では、-Xms に InitServlet の終了時のメモリー使用量の値を設定します。このようにすると、少なくとも、できる限り少ないガーベッジ・コレクションで DEBS の初期化を完了できます。メモリー使用量の値を入手するには、以下のステップを実行します。

  1. -Xms に -Xmx と同じ値を設定します。
  2. Visual Modeler のデプロイメントを開始し、初期化が完了するまで待ちます。
  3. e-commerce サイトのホーム・ページにアクセスします。
  4. テキスト・エディターで **debs.log** ファイルを開き、次のようなログ・エントリーがないか調べます。

     ```
     2003.03.18 ... END Request ... Mem=129380744/388726784/391291344 ...
     ```

  5. **Mem=** に続く最初の数値が、初期化後の現在のメモリー使用量です。-Xms にその数値を設定します。上の例では、値 -Xms128m を使用します。
- -XX:MaxPermSize は、クラスやメソッドなどのシステムのような自己反映オブジェクトに割り振るサイズを制御します。推奨する初期値は 128m です。

  Web アプリケーションの場合、***.jsp** ファイルが ***.java** ファイルに変換されてから、さらに ***.class** ファイルに変換され、このファイルが -XX:MaxPermSize によって指定されたメモリー領域にロードされるため、割り振られた領域はすぐに満杯になります。Java バージョン 1.4.2 から、-XX:+PrintGCDetails を使用して、Permanent Generation と呼ばれるこの領域の詳細をモニターできます。

アプリケーション・サーバーが現在サポートしているものと矛盾するようなメモリー関連の変更は行わないように注意してください。DEBS は、アプリケーション・サーバーと同じ VM で共存する必要があるため、確信が持てない場合は、アプリケーション・サーバーの資料を再確認するか、またはアプリケーション・サーバーのサポート組織に問い合わせてください。例えば、現在のアプリケーション・サーバーの資料に JVM の設定 -server をサポートしていないと明記されているとします。その場合は、-server を設定しません。

最後のトラブルシューティングの手段として、追加の引数なしでまず VM を起動し、引数を 1 つ追加して VM を再起動して結果を確認し、また 1 つ引数を追加して同じことを繰り返すというやり方で良い結果が得られるまで徐々に引数を追加していきます。

その他のパフォーマンスのチューニング

その他のパフォーマンスのチューニングとして、Java ガーベッジ・コレクションのアクティビティー関連の調整や、スレッド、JVM スタック、またはネイティブ構造体またはネイティブ・コードなどのその他の領域のメモリーの設定の調整を行うことができます。Log Analyzer ツールを使用するか、または直接 **debs.log** ファイルを確認して内容を精査し、パフォーマンスの問題領域を特定します。

ガーベッジ・コレクションのアクティビティーの追跡

説明のつかない一時停止が発生する場合は、VM がフル・ガーベッジ・コレクションのために一時停止されている可能性があります。それが起きていることを確認するには、JVM 設定の -verbose:gc を使用して、**debs.log** にガーベッジ・コレクションのイベントを記録できるようにします。ガーベッジ・コレクションのイベントは、次のいずれかのタイプです。

```
[GC 325816K->83372K(776768K), 0.2454258 secs]
[Full GC 267628K->83769K(776768K), 1.8479984 secs]
```

小規模のガーベッジ・コレクションは、1/2 秒未満のはずです。大規模なガーベッジ・コレクションは、3 秒未満のはずです。3 秒を超えるガーベッジ・コレクションは、範囲外条件を示し、調査する必要があります。

ガーベッジ・コレクションを追跡するためのその他の設定として検討できるものを以下に示します。

- JVM-server 設定: この設定は、初期の Java ヒープ設定値の一部を調整して、サーバー環境に適した設定値になるようにします。アプリケーション・サーバーが -server をサポートしない場合以外は、-server 値を設定します。

  -server 設定には、データ・サービスが使用する値を予想外に変更する原因となる JIT (just-in-time) コンパイルのバグに関連する既知の問題があります。その結果、DEBS の初期化に失敗します (InitServlet が失敗します)。特定のメソッドに対して JIT コンパイルを無効にする方法については、担当者に問い合わせてください。

  アプリケーション・サーバーによっては、VM 設定の -server を使用することを推奨しています。特に、-server に対する-XX:NewRatio の値は 2 です (-client 設

定のデフォルト値は 8 です)。-server および -client の設定の詳細については、以下の URL にある Sun の資料を参照してください。

```
http://java.sun.com/docs/books/performance/1st_edition/html/
JPAppHotspot.fm.html#998292
```

```
http://java.sun.com/docs/books/performance/1st_edition/html/
JPAppHotspot.fm.html#998359
```

- -Xincgc 設定: この設定によって、増分ガーベッジ・コレクションが可能になります。Setting -Xincgc を設定すると、フル・ガーベッジ・コレクションによる長い一時停止を削減することができます。この設定を使用する場合は、1 つの大規模のガーベッジ・コレクションに費やす時間をいくつかの小規模のガーベッジ・コレクションにシフトするということに留意してください。このシフトには、通常、約 10% のオーバーヘッド・コストが生じます。
- OutOfMemoryError メッセージが表示される場合は、まず-Xmx の値を大きくする必要がありますが、-Xmx がマシンの物理メモリーの値を超えないようにします。現在のヒープ (新たなオブジェクトが割り振られた) の使用量が -Xmx の値とかけ離れているときに OutOfMemoryError メッセージが表示されるようである場合は、メモリーが割り振られた他の領域が使い果たされた可能性があります。Log Analyzer のレポートを調べて、可能性のある以下の領域を確認します。
  - Due to Classes (クラスが原因): -XX:MaxPermSize=128m の設定を試みます
  - Due to Threads (スレッドが原因): -Xss=512k を使用してスタックの調整を試みます
  - Due to JVM Stacks (JVM スタックが原因): -Xss=512k を使用してスタックの調整を試みます
  - Due to Native data structures (ネイティブ・データ構造体が原因): OS スワップ・サイズの調整を試みます
  - Due to Native codes (ネイティブ・コードが原因): OS スワップ・サイズの調整を試みます

第 7 章 Log Analyzer ツール

Log Analyzer は、Visual Modeler の **debs.log** エントリーの分析に役立つオープン・ソースのツールです。このツールは、重要なパフォーマンス指標であるスレッド、メモリー、要求、およびセッションに加えてユーザーおよび要求のタイプ別にソートした応答時間を視覚化して表示します。

実装環境の日常的なモニタリングの一環として Log Analyzer を使用すると、以下のようなメリットがあります。

- Log Analyzer の日次レポートは、実装環境の信頼性および安定度を向上します。生成されたデータは、潜在的な問題についての早期警告を示すことができ、停止を防止することを可能にします。例えば、毎日の Log Analyzer レポートを使用して、アプリケーション・サーバーが最大に近いメモリー使用量に到達したときにアプリケーション・サーバーを再起動するように予防策を計画することができます。
- Log Analyzer の日次レポートは、現在の問題のトラブルシューティングの基本となるものを提供します。レポートを調べることにより、問題がいつから始まったのか確認し、それを OS のアップグレードなどのイベントと関連付けることができます。
- Log Analyzer の日次レポートは、段階的な改善を行うためのフォーカル・ポイントとなります。Log Analyzer レポートを毎日レビューすることにより、アプリケーション・サーバーの再起動、例外リストのクリーンアップ、ハング状態のスレッドの追跡、または長期に実行されている要求やリソースをかなり使用している要求 (データベースから多数の行を返す要求など) についてのフィードバックの開発者への提供を行う時期を計画するための To Do リストを作成できます。

Log Analyzer ツールを入手するには、担当者に連絡してください。Log Analyzer は、**.jar** ファイルとして提供され、都合のよい場所に保存して解凍できます。Log Analyzer は、DEBS ログ・エントリーの形式が次のような形式であるとみなします。

```
<YYYY.MM.DD HH:MM:SS:mss ThreadName:LogLevel:LogTag:messages>
```

例:

```
2006.10.12 06:00:00:171 Env/http-8580-Processor48:INFO:WrappingFilter ...
```

ログ・ファイルの分析処理はメモリーをかなり使用する可能性があるため、OutOfMemoryError メッセージを防止するためにできる限り多くの JVM メモリーを指定します。例えば、次のようにして Log Analyzer を起動します。

```
java -Xms256m -Xmx512m -jar logAnalyzer-1.1.1-SNAPSHOT-app.jar
```

Log Analyzer の初期画面が表示されます。

次の情報を入力します。

- **Source**: DEBS ログ・ファイルの場所、または複数の DEBS ログ・ファイルが含まれるディレクトリーの絶対パス名を入力します。

- **Input**:
  - 「**DEBS 6.4 or later**」には自動的にチェック・マークが付けられています。リリース 6.4 よりも前の Visual Modeler からのログ・ファイルを分析する場合は、このチェック・ボックスのチェック・マークを外します。
  - 「**WebLogic から (From WebLogic)**」: ログ・ファイルが Oracle WebLogic アプリケーション・サーバーから生成されるように指定するには、このチェック・ボックスをクリックします。
- 「**Output**」: メッセージ・タイプ別にグループ分けされた応答時間のチャートを生成するには、「**Output**」チェック・ボックスをクリックします。
- 「**Output dir**」: レポートの出力先のディレクトリーの絶対パス名を入力します。

「**Start analyzing**」をクリックして、ログ分析処理を開始します。Log Analyzer は、処理の進行中にメッセージを表示し、処理が終了すると指定された出力ディレクトリーにログの分析結果を出力します。

Log Analyzer の日次レポートのセットアップ

このセクションでは、Log Analyzer のレポートを自動的に生成するための手順を説明します。ここで説明する手順では、Ant が移植可能でよく使われ、また優れた資料があるため Ant を使用します。Ant は **http://ant.apache.org** から入手できます。

この手順の目標は、以下のとおりです。

- レポートの生成を毎夜実行するようにクーロン・ジョブをセットアップし、ナビゲーションが容易になるように出力を日付 (年/月/日) ごとに整理する
- スペースの節約のためにできる限りログ・ファイルを圧縮する
- 複数の実装環境からのログ・ファイルを単一のログ・サーバーからホストできるように、容易に繰り返しが可能な方法で自動化をセットアップする

Log Analyzer のレポート生成を自動化するには、以下のものが必要になります。

- Java および Ant
- DEBS ログ・ファイルへの読み取りアクセス権限
- レポート出力ディレクトリー、***<out.dir>*** への書き込みアクセス権限 ***<out.dir>*** の内容には、Web サーバーからアクセスできます。

Log Analyzer の日次レポートの自動化

このタスクについて

以下の手順は、Log Analyzer の日次レポートを自動化するための一般的なワークフローを説明しています。

手順

1. DEBS は、ログ・ファイルをアプリケーション・サーバーまたはサーブレット・コンテナーの **logs** ディレクトリーに出力します。

   例えば、Tomcat デプロイメントの **logs** ディレクトリーは、***<tomcat-home>*\logs** になります。

ログ・ファイルの名前は **debs.log.*n*** になります。ここの *n* は番号です。例えば、debs.log.1、debs.log.2 など

2. ログ・ディレクトリーからのすべてのログ・ファイルを連結して一時ファイルに出力するために、クーロン・ジョブを毎日 (おそらく早朝) 実行するようにセットアップします。
3. 一時ファイルから、ディレクトリー命名パターンの year/month/day/log.suffix を使用して、昨日のログ・エントリーを Log Analyzer の出力ディレクトリーに抽出します。
4. **year/month/day/log.suffix** ファイルは、スペースを節約するために **gzip** を使用してさらに圧縮されます。
5. Log Analyzer を起動して、**year/month/day/log.suffix.gz** ファイルを解析し、レポートを **year/month/day/html/** ディレクトリーに生成します。

Log Analyzer の日次レポートの設定

手順

1. まだの場合は、担当者に連絡して、以下のファイルを入手します。
   - Log Analyzer **.jar** ファイル
   - **logAnalyzer-daily.xml**
   - logAnalyzer-daily.properties
2. Log Analyzer ファイルを一時的な場所に保存します。
3. 構成値については、38 ページの『Log Analyzer の構成』 を参照してください。
4. 次のコマンドを実行して、Log Analyzer の日次レポートを実行します。

   ```
   ant -Dproperties.file.name="logAnalyzer-daily.properties"  -f logAnalyzer-daily.xml
   ```

5. 出力を調べます。出力は次のようなところにあります。

   ```
   sites/default/app-server/logAnalyzer-out.d/dailySplit/YYYY/MM/DD/html/index.html
   ```

推奨するディレクトリー・レイアウト

以下の図は、推奨する Log Analyzer のディレクトリー・レイアウトを示しています。このレイアウトは、複数のサイトからのログ・ファイルをホストする予定である場合に、特に推奨します。

```
# where to keep the log analyzer jar file
bin/
   logAnalyzer-1.1.1-SNAPSHOT-app.jar

# ant script
logAnalyzer-daily.xml

# sites data
sites/
     site1/
          ...
     site2/
          ...
     siteN/
          logAnalyzer-daily.properties
          app-server/
                    logs/
                         debs.log
                         debs.log.1
                         debs.log.2
```

サイト情報は、各サイトの情報が含まれる、**sites** ディレクトリー下に保持されます。サイト・ディレクトリーの名前は任意の固有の文字列にすることができます。上の例では、site*n* が使用されています。ここの *n* は番号です。つまり、site1、site2 などです。

各サイト・ディレクトリーには、サイト固有の設定値が含まれる **logAnalyzer-daily.properties** ファイルが含まれています。

各サイトのログ・ファイルは、**site*n*/app-server/logs/** ディレクトリーに保持されます。

sites ディレクトリーは読み取り専用です。出力は、**site*n*/app-server/logAnalyzer-out.d** ディレクトリーに書き込まれます。

上記のレイアウトを使用すると、サイト名だけでクーロン・ジョブを起動できます。例えば、サイト名が bbfb-01 である場合は、次のようになります。

```
# Tell Ant to set the site.name and use a build script name:
# logAnalyzer-daily.xml
ant -Dsite.name=bbfb-01 -f logAnalyzer-daily.xml
```

logAnalyzer-daily.xml の名前を **build.xml** に変更すると、**-f logAnalyzer-daily.xml** 引数を省略することができます。例えば、サイトの名前が bbfb-01 である場合は、次のようになります。

```
ant -Dsite.name=bbfb-01
```

Log Analyzer の構成

デプロイメント固有の設定値は、プロパティー・ファイルに設定されます。デフォルトのプロパティー・ファイルは、**sites/${site.name}/logAnalyzer-daily.properties** です。次のようにしてコマンド・ラインでプロパティー・ファイル名を設定することもできます。

```
ant -Dproperties.file.name="path_to_file.properties" ...
```

以下は、**logAnalyzer-daily.properties** ファイル内の構成プロパティーを示しています。

- log.dir: DEBS ログ・ファイルが含まれるディレクトリーの場所への絶対パス。例:

```
# default is ./logs
log.dir=/home/hle/tomcat/logs
```

- out.dir: 生成されたレポートを書き込む場所。例:

```
# default is logAnalyzer-out.d
out.dir=/home/hle/public_html/logAnalyzer-out.d
```

- **logAnalyzer.jar**: logAnalyzer **.jar** ファイルの場所。例:

```
# default is ./logAnalyzer-1.1.1-SNAPSHOT-app.jar
logAnalyzer.jar=target/logAnalyzer-1.1.1-SNAPSHOT-app.jar
```

- is.weblogic: ログ・ファイルが WebLogic によって生成された場合は true。例:

```
# default is false
is.weblogic=true
```

- genChart.perMessageType: messageType チャートの生成をスキップする場合は false。例:

```
# default is true
genChart.perMessageType=false
```

- log.prefix: DEBS ログのプレフィックス。これを変更する必要はほとんどありません。例:

```
# default is debs.log
log.prefix=Midwest.log
```

- target.date.offset: target.date を自動設定します。デフォルトは 1 です。これは昨日を意味します。例えば、次のように target.date.offset に 7 を設定して、1 週間前のログ・ファイルを抽出します。

```
# default is yesterday: 1
target.date.offset=7
```

- target.date: 当日のログ・エントリーに処理を限定します。この設定が使用される最も可能性の高い用途としては、ログ・ファイルの古いセットの手動による再生成があげられます。例:

```
# default is yesterday (auto-evaluated)
target.date=2006/07/24
```

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ラ行]

D

J

L

[特殊文字]

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

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

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

〒103-8510

東京都中央区日本橋箱崎町19番21号

日本アイ・ビー・エム株式会社

法務・知的財産

知的財産権ライセンス渉外

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

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

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

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

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

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA 95141-1003

U.S.A.

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

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

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

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

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

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

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

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

著作権使用許諾:

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

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

© IBM 2011。このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. 2011。

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

商標

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

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

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

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

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

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

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

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

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

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

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

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

Printed in Japan

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