付録 F

patterns & practices の Enterprise Library

概要

この付録では、patterns & practices の Enterprise Library について説明します。また、お使いのアプリケーションで Enterprise Library を使用して、ログ記録、例外ハンドル、データ アクセスなどの横断的関心事をすばやく簡単に実装する方法を紹介します。

Enterprise Library の目標

Enterprise Library は次の特性を実現することを目的としています。

- **一貫性**: Enterprise Library のすべてのアプリケーション ブロックには、一貫性のある設計パターンと実装の手法が用意されています。
- **拡張性**: Enterprise Library のすべてのアプリケーション ブロックには、明確な拡張ポイントが用意されており、開発者は独自のコードを追加して、アプリケーション ブロックの動作をカスタマイズできます。
- **利便性**: Enterprise Library では、グラフィカルな構成ツール、簡単なインストール手順、わかりやすく包括的なドキュメントやサンプルなど、便利な機能を数多く提供しています。
- **統合**: Enterprise Library のアプリケーション ブロックは、連携して機能するように設計されており、連携の確認テストも行われています。また、アプリケーション ブロックは個別に使用することもできます。

Enterprise Library の機能

Enterprise Library には次の機能が用意されています。

- ログ記録、例外ハンドル、検証、データ アクセスなどの横断的関心事に関するソリューションの実装に使用できる再利用可能なコードで構成されたアプリケーション ブロック。
- 簡単に Enterprise Library のアプリケーション ブロックをアプリケーションに追加したり、構成情報を指定できる構成ツール。構成ツールには、スタンドアロンの構成エディターや、Visual Studio と統合される構成ツールがあります。
- ライブラリやアプリケーション ブロックのあらゆる場所で使用されているシリアル化などの一般的なユーティリティの機能。これは開発者が独自に記述したコードで使用できます。
- 開発者や管理者が、実行時にアプリケーション ブロックの動作やパフォーマンスを監視できるインストルメンテーション機能。
- Enterprise Library のソース コードを作成したり、アセンブリを適切な場所にコピーするバッチ ファイル。
- Enterprise Library で公開されるイベントやパフォーマンス カウンターのインストルメンテーションをインストールするユーティリティ。
- Enterprise Library の例やクイック スタートで使用されるサンプル データベースを作成するユーティリティ。
- 一連のクイック スタート アプリケーションは、各アプリケーション ブロックに対応し、アプリケーション ブロックの使用方法を示します。また、各アプリケーション ブロックの一般的なシナリオを実装しており、ガイダンス ドキュメントの関連セクションへのリンクが提供されています。
- Enterprise Library の完全なソース コード。Visual Studio のプロジェクトや単体テストも用意されており、開発者はこれを使用して、ライブラリやアプリケーション ブロックを拡張したり変更することができます。開発者は、単体テストを実行したり、新しいテストを作成して、アプリケーションが設計要件を満たしていることを確認できます。

アプリケーション ブロック

次の表に、開発者が一般的なエンタープライズ開発の課題を解決するのに役立つように設計された、アプリケーション ブロックの説明を示します。

アプリケーション ブロック	説明
Caching Application Block	開発者がローカル キャッシュをアプリケーションに組み込めるようにします。メモリ内キャッシュや、必要に応じて、データベースまたは分離ストレージとして機能するバッキング ストアをサポー

	トします。キャッシュ データを取得、追加、および削除するために必要なすべての機能を提供し、構成可能な有効期限と清掃のポリシーをサポートします。
Cryptography Application Block	開発者が暗号化機能をアプリケーションに組み込む方法を簡略化します。アプリケーションでは、このアプリケーション ブロックを使用して、情報を暗号化する、データからハッシュを作成する、ハッシュ値を比較してデータが変更されていないことを確認するなど、さまざまなタスクを実行できます。
Data Access Application Block	データを表示するために読み取る、アプリケーション レイヤーを通じてデータを渡す、変更されたデータをデータベース システムに送信するなど、一般的なデータ アクセス機能を実装する開発タスクを簡略化します。ストアド プロシージャとインライン SQL がサポートされています。また、使いやすいクラスの形式で頻繁に使用する ADO.NET の機能にアクセスできます。
Exception Handling Application Block	このアプリケーション ブロックを使用して、開発者やポリシー上の意思決定者は、エンタープライズ アプリケーションのすべてのアーキテクチャ レイヤーで発生する例外をハンドルする一貫した方針を作成できます。例外情報をログに記録したり、元の例外を別の例外に置き換えて機密情報が公開されないようにしたり、元の例外を別の例外内にラップすることで例外のコンテキスト情報を管理したりできます。
Logging Application Block	一般的なログ記録の機能を簡単に実装できます。Windows イベント ログ、電子メール メッセージ、データベース、Windows メッセージ キュー、テキスト ファイル、WMI イベント、または独自の場所に情報を書き込むことができます。
Policy Injection Application Block	オブジェクト インスタンスにポリシーを自動的に適用することで、開発者が、横断的関心事をより適切に管理したり、横断的関心事をできる限り分離したり、動作をカプセル化したりするのに役立ちます。開発者は、構成を通じて、または対象のクラスの個々のメンバーに属性を適用して、クラスとそのメンバーを対象とした一連のポリシーを定義します。
Security Application Block	開発者が、アプリケーションに一般的な承認関連の機能を実装したり、ユーザーの承認と認証に関するデータをキャッシュするのに役立ちます。Microsoft .NET Framework 2.0 の機能を併用すると、開発者は、一般的なセキュリティ関連の機能を簡単に実装できます。
Unity Application Block	コンストラクター、プロパティ、およびメソッド呼び出しの挿入を

	サポートする、軽量で拡張可能な依存関係の挿入 (DI) のコンテナーを提供します。開発者は、このコンテナーを Enterprise Library と併用して、Enterprise Library のオブジェクトや独自のカスタム ビジネス オブジェクトを生成したり、コンテナーをスタンドアロンの DI メカニズムとして使用したりできます。
Validation Application Block	開発者が、構造化され保守が容易な検証シナリオをアプリケーションに実装できる、便利な機能を提供します。null 文字列、数字の範囲の検証コントロールなど、.NET Framework のデータ型を検証するための検証コントロールのライブラリが用意されています。また、複合検証コントロールも用意されており、ルール セットをサポートしています。

Caching Application Block

Caching Application Block を使用して、メモリ内キャッシュや、必要に応じて、データベースまたは分離ストレージとして機能するバッキング ストアを使用するローカル キャッシュをアプリケーションに組み込むことができます。キャッシュ データを取得、追加、および削除するために必要なすべての機能を提供し、構成可能な有効期限と清掃のポリシーをサポートします。また、分散キャッシュなどの機能をサポートするために、プラグ可能なプロバイダーを独自に作成したり、サード パーティ製のプロバイダーを使用して、アプリケーション ブロックを拡張することもできます。キャッシュを使用すると、多くのアプリケーションのシナリオでパフォーマンスと効率が大幅に向上します。

主要なシナリオ

Caching Application Block は、次のような状況に直面した場合に適しています。

- 静的なデータまたはほとんど変更されないデータに繰り返しアクセスしている。
- 作成、アクセス、または移動で負荷の高いデータ アクセスを行っている。
- サーバーなどのソースが使用できない場合でも、データは常に利用できる必要がある。

使用すべき状況

Caching Application Block は、高度なパフォーマンスとスケーラビリティを実現するように最適化されています。また、スレッド セーフであり、例外セーフでもあります。このアプリケーション ブロックは、拡張して、独自の有効期限ポリシーやバッキング ストアを含めることができます。アプリケーションとキャッシュが同じシステムに存在するという、最も一般的なデータ キャッシュで機能するように設計されています。つまり、キャッシュはローカルに配置して、そのアプリケーション専用のキャッシュとして使用される必要があります。ア

プリケーションが以上のガイドラインで動作する場合、このアプリケーション ブロックは次の要件に対処するのに最適です。

- 異なるアプリケーション環境のキャッシュ機能について、一貫性のある (使用しているキャッシュ ストアによって変わることがない) シンプルなインターフェイスと実装が必要である。たとえば、開発者は、インターネット インフォメーション サービス (IIS)、エンタープライズ サービス、およびスマート クライアント環境でホストされているアプリケーション コンポーネントにキャッシュを実装するために同様のコードを作成できます。また、すべての環境に同様のキャッシュ構成オプションが存在します。
- 構成可能で永続的なバッキング ストアが必要である。このアプリケーション ブロックでは、分離ストレージやデータベースとして機能するバッキング ストアをサポートしています。開発者は、追加のバッキング ストアのプロバイダーを作成し、構成設定を使用してアプリケーション ブロックに追加できます。また、アプリケーション ブロックでは、キャッシュ項目のデータを対称的に暗号化してから、バッキング ストアに配置できます。
- キャッシュ構成の設定を変更する際に、アプリケーションのソース コードを変更する必要がないようにする。開発者は、まず、名前付きキャッシュを 1 つ以上使用するコードを作成します。システム オペレーターと開発者は、Enterprise Library の構成ツールを使用して、異なる名前が付けられたキャッシュをそれぞれ構成することができます。
- キャッシュ項目に、絶対時間、相対時間、拡張された時刻の形式 (たとえば、毎晩真夜中になど)、ファイルの依存関係、有効期限なしなど、有効期限を設定する必要がある。
- アプリケーション ブロックのソース コードを変更して、拡張性を高めたり、カスタマイズすることができます。
- 1 つのアプリケーションで、複数の種類のキャッシュ ストアを (異なるキャッシュ マネージャーを通じて) 使用する必要がある。

Caching Application Block は次の種類のアプリケーションで使用できます。

- Windows フォーム
- コンソール アプリケーション
- Windows サービス
- COM+ サーバー
- ASP.NET Web アプリケーションまたは Web サービス (ASP.NET キャッシュに含まれていない機能が必要な場合)

考慮事項

Caching Application Block を使用する際には、次の事項を考慮します。

- このアプリケーション ブロックは、単一のアプリケーション ドメインに配置する必要があります。各アプリケーション ドメインには、バッキング ストアを使用するかどうかにかかわらず、1 つまたは複数のキャッシュ ストアを含められます。
- キャッシュ ストアは、複数のアプリケーション ドメインで共有できません。
- このアプリケーション ブロックでは、バッキング ストアにキャッシュされたデータは暗号化できますが、メモリ内にキャッシュされたデータの暗号化はサポートしていません。
- このアプリケーション ブロックでは、改ざん防止対策 (署名やキャッシュ内の項目の確認) をサポートしていません。

Cryptography Application Block

Cryptography Application Block を使用すると、情報を暗号化する、データからハッシュを作成する、ハッシュ値を比較してデータが変更されていないことを確認するなどの暗号化機能を簡単に組み込むことができます。

主要なシナリオ

Cryptography Application Block は、次のような状況に直面した場合に適しています。

- すばやく簡単に情報の暗号化と復号化を行っている。
- すばやく簡単にデータからハッシュを作成している。
- ハッシュ値を比較して、データが変更されていないことを確認している。

使用すべき状況

Cryptography Application Block は次の要件に対処するのに最適です。

- 標準的なデータの暗号化、復号化、およびハッシュのタスクを実行するために作成する、ひな型のコードの要件を減らす必要がある。
- アプリケーションと社内の両方で、一貫した暗号化のノウハウを保持する必要がある。
- さまざまな領域の機能で一貫したアーキテクチャのセキュリティ モデルを使用することで、開発者が習得しなければならないことを最小限に抑える必要がある。
- 暗号化プロバイダーの実装を追加したり拡張する必要がある。
- カスタマイズ可能な主要な保護モデルが必要である。

考慮事項

Cryptography Application Block を使用する際には、次の事項を考慮します。

- このアプリケーション ブロックでは、暗号化と復号化の両方に同じキーを使用する対称的なアルゴリズムのみサポートします。
- このアプリケーション ブロックでは、暗号化キーとキーの格納は自動的に管理しません。

Data Access Application Block

Data Access Application Block では、データを表示するために読み取る、アプリケーション レイヤーを通じてデータを渡す、変更されたデータをデータベース システムに送信するなど、多くの一般的なデータ アクセスのタスクを簡略化します。このアプリケーション ブロックでは、ストアド プロシージャとインライン SQL がサポートされています。また、使いやすいクラスの形式で頻繁に使用する ADO.NET の機能にアクセスできます。

主要なシナリオ

Data Access Application Block は、次のような状況に直面した場合に適しています。

- 複数のデータ行を取得するのに、DataReader または DataSet を使用している。
- コマンドを実行して、出力パラメーターまたは単一の値の項目を取得している。
- 1 つのトランザクションで複数の操作を実行している。
- SQL Server から XML データを取得している。
- DataSet オブジェクトに含まれるデータを使用してデータベースを更新している。
- データベース プロバイダーの実装を追加したり拡張したりしている。

使用すべき状況

Data Access Application Block は次の要件に対処するのに最適です。

- 開発者が ADO.NET で提供される機能をベスト プラクティスに従って使用する際に、使いやすさと利便性が求められる。
- 標準的なデータ アクセスのタスクを実行するために作成する、ひな型のコードの要件を減らす必要がある。
- アプリケーションと社内の両方で、一貫したデータ アクセスのノウハウを保持する必要がある。

- 使用するデータベースの種類を構成を通じて変更したり、開発者がアプリケーションを異なる種類のデータベースに移植するときに、記述する必要があるコードの量を簡単に減らせる必要がある。
- 開発者が、データベースの種類ごとに異なるプログラミング モデルを学習しないで済むようにする必要がある。

考慮事項

Data Access Application Block を使用する際には、次の事項を考慮します。

- Data Access Application Block は ADO.NET の機能を補完するものであり、置き換わるものではありません。アプリケーションで、特殊な方法でデータを取得したり、特定のデータベース固有の機能を使用する必要がある場合は、ADO.NET を直接使用することを検討します。

Exception Handling Application Block

Exception Handling Application Block を使用すると、アプリケーションのすべてのアーキテクチャ レイヤーで発生する例外を処理する一貫した方針をすばやく簡単に設計したり実装したりできます。例外情報をログに記録したり、元の例外を別の例外に置き換えて機密情報が公開されないようにしたり、元の例外を別の例外内にラップすることで例外のコンテキスト情報を管理したりできます。

主要なシナリオ

開発者は、Exception Handling Application Block を使用して、アプリケーション コンポーネントの catch ステートメントに含まれているロジックを再利用可能な例外ハンドラーとしてカプセル化できます。このアプリケーション ブロックは、次のような要件に直面した場合に適しています。

- 例外をラップする。Wrap ハンドラーを使用して、新しい例外で例外をラップします。
- 例外を置き換える。Replace ハンドラーを使用して、元の例外を別の例外に置き換えます。
- 例外をログに記録する。Logging ハンドラーを使用して、例外情報をメッセージ、スタック トレースなどの形式にして、Enterprise Library の Logging Application Block に渡して公開できるようにします。
- WCF サービス境界で例外を隠ぺいする。Windows Communication Foundation (WCF) サービス境界で使用するために設計された Fault Contract Exception ハンドラーを使用して、例外から新しいエラー コントラクトを生成します。

- 例外を伝播して、わかりやすいメッセージを表示し、ユーザーに通知して、サポート スタッフを支援する。このアプリケーション ブロックのハンドラーを組み合わせて使用して、特定の例外の種類を処理したり、必要に応じて例外を再度スローします。
- 例外メッセージをローカライズする。ハンドラーと構成を使用して、ローカライズした例外メッセージ テキストを指定します。

使用すべき状況

Exception Handling Application Block は次の要件に対処するのに最適です。

- サービス インターフェイス境界だけでなく、アプリケーションのすべてのアーキテクチャ レイヤーで例外ハンドルをサポートする必要がある。
- 例外ハンドル ポリシーを、構成を通じて管理者レベルで定義したり管理する必要がある。また、アプリケーション ブロックのコードを変更することなく、例外ハンドルを管理するルールを保守および変更できる必要がある。
- 例外情報をログ記録する機能、元の例外を別の例外に置き換えて機密情報が公開されないようにする機能、元の例外を別の例外でラップして例外のコンテキスト情報を管理する機能など、一般的に使用される例外ハンドルの機能を利用する必要がある。
- 例外情報をログに記録し、元の例外を別の例外に置き換えるなど、例外に対して必要な対応をするために、例外ハンドラーを組み合わせる必要がある。
- アプリケーション内の複数の場所とアプリケーション間でハンドラーを使用できるように、一貫した方法で例外ハンドラーを呼び出す必要がある。
- 例外ハンドラーの実装を追加したり拡張する必要がある。
- 例外をログに記録するだけでなく、ポリシーを通じてハンドルする必要がある。

Logging Application Block

Logging Application Block を使用すると、Windows イベント ログ、電子メール メッセージ、データベース、Windows メッセージ キュー、テキスト ファイル、WMI イベント、または独自の場所に情報を書き込むなど、一般的なログ記録の機能を簡単に実装できます。

主要なシナリオ

Logging Application Block は、次のような状況に直面した場合に適しています。

- イベント情報を Windows イベント ログ、電子メール メッセージ、データベース、メッセージ キュー、テキスト ファイル、Windows Management Instrumentation (WMI) イベント、または独自の場所に設定してログに記録している。
- テンプレートを使用して、イベントのコンテキスト情報を変更したり書式設定をしている。
- アプリケーションの動作を追跡したり、イベント情報を組み合わせるために使用できる ID を提供している。
- フラット ファイルへのアクセスを制限するためにアクセス制御リスト (ACL) を使用したり、ログ情報を暗号化するカスタム フォーマッターを作成して、機密情報に不当にアクセスされないようにしている。

使用すべき状況

Logging Application Block は次の要件に対処するのに最適です。

- アプリケーションと社内の両方で、一貫したログ記録のノウハウを保持する必要がある。
- 一貫したアーキテクチャ モデルを使用することで、開発者が習得しなければならないことを最小限に抑える必要がある。
- カスタム コードまたはひな型のコードを繰り返し記述することなく、一般的なアプリケーションのログ記録に関するタスクを解決できる実装を提供する必要がある。
- ログ記録の実装や対象を追加したり拡張する必要がある。

考慮事項

Logging Application Block を使用する際には、次の事項を考慮します。

- このアプリケーション ブロックのログ記録のフォーマッターでは、ログ記録情報が暗号化されません。
- トレース リスナーの出力先は、クリア テキストでログ記録情報を受け取ります。
- Logging Application Block のトレース リスナーには、部分的に信頼された状態で実行するとエラーが発生するものがあります。

Policy Injection Application Block

Policy Injection Application Block で使用できる、オブジェクト インスタンスにポリシーを自動的に適用するメカニズムを利用することで、開発者は横断的関心事をより適切に管理したり、横断的関心事をできる限り分離

したり、動作をカプセル化したりできます。開発者は、Policy Injection Application Block の構成を通じて、または対象のクラスの個々のメンバーに属性を適用して、クラスとそのメンバーを対象とした一連のポリシーを定義します。

主要なシナリオ

Policy Injection Application Block は、次のような状況に直面した場合に適しています。

- 操作に依存関係がないことから多くのメリットを得て、できる限り再利用できるようにするため、カプセル化や分離が必要なオブジェクトでアプリケーションを構築している。
- 開発者、オペレーター、および管理者が、通常、アプリケーションのコードや再コンパイルを変更することなく、構成を通じてインターセプト ポリシーを作成、変更、削除、および微調整できるようにしている。このようにすることで、コードでエラーが発生する可能性が減少し、バージョン管理が簡略化して、ダウンタイムを短縮できます。
- 既存のオブジェクト インスタンスを再利用している。このようにすると、新しいオブジェクト インスタンスを作成して、プロパティを設定したりメソッドを呼び出して準備をするためのコードを記述する必要性を低減しながら、クラスの特定のメンバーまたはすべてのメンバーに対して、ハンドラーのパイプラインを使用することができます。
- アプリケーションで、ログ記録、検証、承認、インストルメンテーションなど、一般的なタスクを実行するために必要な作業や開発者が記述しなければならないコードを最小限に抑えている。

使用すべき状況

Policy Injection Application Block は次の要件に対処するのに最適です。

- 新しいアプリケーションや既存のアプリケーション (特に Enterprise Library の機能を既に使用しているアプリケーション) に簡単に実装できる既製のソリューションが必要である。
- 一般的な機能 (ログ記録、検証など) にアクセスする必要があるオブジェクトの独立性に影響するおそれのある横断的関心事を管理する必要がある。
- 開発者や管理者が、一般的なタスクを実行するハンドラーや、カスタムの機能を追加するハンドラーを追加または削除して、構成を通じてアプリケーションのオブジェクトの動作を構成できるようにする必要がある。
- 開発者が、Enterprise Library Core の機能やエンタープライズ アプリケーションで一般的に必要なタスクを実装している個々のアプリケーション ブロックの機能を、簡単に利用できる必要がある。
- 開発にかかる時間とコストを削減したり、一般的な共有のタスクやサービスを使用する複雑なアプリケーションで、バグを最小限に抑える必要がある。

考慮事項

Policy Injection Application Block を使用する際には、次の事項を考慮します。

- このアプリケーション ブロックでは、コードを直接メソッドに挿入するのではなく、インターセプトを使用して前処理と後処理のハンドラーのみを有効にします。
- クラス コンストラクターではインターセプトを使用できません。
- アプリケーション ブロックが処理要件を可能な限り最小限に抑えるように設計されていますが、すべてのインターセプト技術と同様、アプリケーションには追加の要件が課されます。
- Call ハンドラーは呼び出しメッセージの情報にしかアクセスできず、内部状態を管理することはできません。
- ポリシーの挿入は、対象のクラスのパブリック メンバーに対してのみ実行できます。

Security Application Block

Security Application Block を使用すると、ユーザーの承認と認証に関するデータをキャッシュしたり、Microsoft .NET Framework のセキュリティ機能と統合するなど、一般的な承認関連の機能を簡単に実装できます。

主要なシナリオ

Security Application Block は、次のような状況に直面した場合に適しています。

- 承認の実行に使用する、セキュリティ関連の資格情報をキャッシュしている。
- 認証されたユーザーに関する一時的なトークンを取得し、トークンを使用してユーザーを認証している。
- ユーザー セッションを終了している (トークンを有効期限切れにしている)。
- タスクを実行する際に、ユーザーに権限があるかどうかを判断している。

使用すべき状況

Security Application Block は次の要件に対処するのに最適です。

- 資格情報のキャッシュ、認証の確認など、標準的なセキュリティ関連のタスクを実行するのに使用する、ひな型のコードの要件を減らす必要がある。
- アプリケーションと社内の両方で、一貫したセキュリティのノウハウを保持する必要がある。

- 利用するさまざまな領域の機能で一貫したアーキテクチャ モデルを使用することで、開発者が習得しなければならないことを最小限に抑える必要がある。
- セキュリティ プロバイダーのカスタム実装を使用する必要がある。

考慮事項

Security Application Block を使用する際には、次の事項を考慮します。

- キャッシュされるセキュリティ関連情報の既定のストアは、Caching Application Block です。Caching Application Block では、バッキング ストアのキャッシュ データを暗号化するように構成できますが、メモリ内のキャッシュ データの暗号化はサポートしていません。このことがアプリケーションにとって大きな脅威となる場合は、メモリ内の暗号化をサポートしている別のカスタムのキャッシュ ストアのプロバイダーを使用できます。
- 承認マネージャーは部分的に信頼された状態ではサポートされません。

Unity Application Block

Unity Application Block は、オブジェクト インターセプト、コンストラクターの挿入、プロパティの挿入、およびメソッド呼び出しの挿入をサポートする、軽量で拡張可能な依存関係の挿入のコンテナーです。また、Enterprise Library と併用して、Enterprise Library オブジェクトと独自のカスタム ビジネス オブジェクトを生成できます。

主要なシナリオ

Unity Application Block は、次のような状況に直面した場合に適しています。

- コンストラクター、プロパティ、およびメソッド呼び出しの挿入をサポートし、オブジェクト インスタンスの有効期限を管理できるコンテナーを通じて、依存関係の挿入を実行している。
- 他のオブジェクトやクラスと依存関係があり、その依存関係が複雑であったり抽象化を必要としているクラスに対して、依存関係の挿入を実行している。
- 実行時に依存関係を構成したり変更している。
- Web アプリケーションのポスト バック間でコンテナーをキャッシュしたり保持している。

使用すべき状況

Unity Application Block は次の要件に対処するのに最適です。

- オブジェクトを簡単に作成できる必要がある (特に階層構造になっているオブジェクトの構造や依存関係を簡略化する必要がある)。このようにすると、アプリケーション コードも簡略化されます。
- 横断的関心事の管理を簡略化するため、実行時または構成で依存関係を指定して、要件を抽象化する必要がある。
- コンポーネントの構成をコンテナーで行うことで、柔軟性を向上する必要がある。
- クライアントがコンテナーを格納したりキャッシュしたりできる、サービスの場所に関する機能が必要である。これは特に ASP.NET Web アプリケーションで役に立ち、開発者は ASP.NET セッションまたはアプリケーションでコンテナーを保持できます。

Unity Application Block は、次のような状況では使用しないようにします。

- オブジェクトとクラスが他のオブジェクトやクラスと依存関係がないか、依存関係が非常に単純で抽象化の必要がない。

考慮事項

Unity Application Block を使用する際には、次の事項を考慮します。

- 依存関係の挿入がパフォーマンスにわずかに影響する可能性があります。
- 単純な依存関係しか存在しない場合に依存関係の挿入を使用すると、複雑さが増す可能性があります。

Validation Application Block

Validation Application Block では、属性とルール セットを使用する構造化されて管理しやすい検証メカニズムを実装したり、多数のアプリケーション インターフェイスのテクノロジと統合するための、さまざまな機能を提供しています。

主要なシナリオ

Validation Application Block は、次のような状況に直面した場合に適しています。

- フィールド、プロパティ、および入れ子になったオブジェクトを検証したり、悪意のあるデータがアプリケーションに挿入されないようにする、構造化されて管理しやすい検証コードを実装しています。
- ビジネス ルールを強化して、ユーザー入力に応答を提供している。

- 同じアプリケーション内で同じ規則を使用してデータを複数回検証している。
- さまざまな構築済みの検証コントロールを組み合わせて、複雑なシナリオやさまざまな機能をサポートしている。

使用すべき状況

Validation Application Block は次の要件に対処するのに最適です。

- ASP.NET、Windows フォーム、および WCF アプリケーションの標準的な .NET のデータ型の大半について、一貫した検証のノウハウを保持する必要がある。
- 構成、属性、およびコードを使用して、検証規則を作成する必要がある。
- 複数のルール セットを、同じクラスやクラスのメンバーと関連付ける必要がある。
- オブジェクトを検証するときに、1 つ以上のルール セットを適用したり、ビジネス検証ロジックを再利用する必要がある。

考慮事項

Validation Application Block を使用する際には、次の事項を考慮します。

- ASP.NET、Windows フォームなどの一部のテクノロジでは、組み込みの検証機能を使用できます。そのため、これらのテクノロジにのみ検証ロジックを適用する必要がある場合、その検証ロジックを再利用する必要がない限り、このアプリケーション ブロックを使用する必要はありません。
- WCF アプリケーションと XML データを使用する他のアプリケーションでは、XML スキーマを使用して XML レベルでメッセージを検証できます。これらのテクノロジにのみ検証ロジックを適用する必要がある場合、その検証ロジックを再利用する必要がない限り、このアプリケーション ブロックを使用する必要はありません。
- 少しのオブジェクトしか検証する必要がないという非常に単純なシナリオでは、アプリケーション ブロックを追加して、オーバーヘッドを負うことはお勧めしません。

関連情報

Web リソースに簡単にアクセスするには、http://www.microsoft.com/architectureguide (英語) でオンライン版の参考文献を参照してください。

- Enterprise Library (http://msdn.microsoft.com/en-us/library/cc467894.aspx、英語)

- The Caching Application Block
  (http://msdn.microsoft.com/en-us/library/cc511588.aspx、英語)
- The Cryptography Application Block
  (http://msdn.microsoft.com/en-us/library/cc511721.aspx、英語)
- The Data Access Application Block
  (http://msdn.microsoft.com/en-us/library/cc511547.aspx、英語)
- Exception Handling Application Block
  (http://msdn2.microsoft.com/en-us/library/aa480461.aspx、英語)
- The Logging Application Block
  (http://msdn.microsoft.com/en-us/library/cc511708.aspx、英語)
- The Policy Injection Application Block
  (http://msdn.microsoft.com/en-us/library/cc511729.aspx、英語)
- The Security Application Block
  (http://msdn.microsoft.com/en-us/library/cc511928.aspx、英語)
- The Unity Application Block
  (http://msdn.microsoft.com/en-us/library/cc511654.aspx、英語)
- The Validation Application Block
  (http://msdn.microsoft.com/en-us/library/cc511802.aspx、英語)
- Microsoft Enterprise Library Frequently Asked Questions
  (http://www.codeplex.com/entlib/Wiki/View.aspx?title=EntLib%20FAQ、英語)