Sitecore CMS 6.4
プレゼンテーション コンポーネント リファレンス

CMS 管理者、アーキテクト、開発者のための概念の概要

目次

Chapter 1

イントロダクション

この文書では Sitecore のレイアウト エンジンで使用されるプレゼンテーション コンポーネントについて説明します。これにはレイアウト、サブレイアウト、レンダリング、プレースホルダーを含みます。CMS のアーキテクト、開発者、管理者には Sitecore ソリューションを実装する前にこの文書をお読みいただくことを推奨します。[1] この文書を活用して、開発、保守、管理のコストを最小化するためのプレゼンテーション コンポーネントとその技術を習得することができます。

この文書ではまずプレゼンテーション コンポーネントの種類を説明します。これにはレイアウト、サブレイアウト、レンダリングがあります。次にレイアウト エンジンがプレゼンテーション コンポーネントを組み立てて HTTP 要求に応答する方法 (デバイスとレイアウト詳細を含む) を説明します。さらにプレゼンテーション コンポーネントの出力をキャッシュする方法を説明し、また各コンポーネントにプレゼンテーション技術を選択する際の考慮点を説明します。

この文書には次の章があります。

- イントロダクション
- プレゼンテーション コンポーネント
- 要求の処理
- 出力のキャッシュ

[1] この文書で説明されている機能の使用法については、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20Cookbook.aspx から『プレゼンテーション コンポーネント クックブック』を参照してください。XSL レンダリングの実装方法に関する詳細については、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20XSL%20Reference.aspx から『プレゼンテーション コンポーネント XSL リファレンス』を参照してください。.NET レンダリング コンポーネントに関する詳細については、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20API%20Cookbook.aspx から『プレゼンテーション コンポーネント API クックブック』を参照してください。

- プレゼンテーション技術の選択

Chapter 2

プレゼンテーション コンポーネント

まず Sitecore のレイアウト エンジンの概要を説明し、次にレイアウト エンジンが使用するプレゼンテーション コンポーネントの種類を説明します。これにはレイアウト、サブレイアウト、レンダリング、プレースホルダーがあります。最後に Sitecore のユーザー インターフェースが使用する XML プレゼンテーション コンポーネントを説明します。

この章には次のセクションがあります。

- レイアウト エンジンの概要
- レイアウト (ASP.NET .aspx Web フォーム)
- サブレイアウト (ASP.NET .ascx Web ユーザー コントロール)
- レンダリング
- プレースホルダー
- Sitecore ユーザー インターフェース プレゼンテーション コンポーネント

2.1 レイアウト エンジンの概要

Sitecore のレイアウト エンジンは Web クライアントがリソースを要求したときに、ASP.NET Web アプリケーション サーバーを拡張し、コンテンツをプレゼンテーション ロジックと動的にマージします。レイアウト エンジンは、Web クライアントから要求された URL に対応するコンテキスト アイテムで定義されたレイアウト詳細に従って、データベースのコンテンツをファイルのコードと動的にマージします。レイアウト エンジンは HTTP 要求を元に、コンテキストの言語、デバイス タイプ、その他の基準を決定します。プレゼンテーション コンポーネントは Sitecore のコンテキストに基づいて、異なる出力を生成します。コンテキスト アイテムに関する詳細は「コンテキスト アイテム」のセクションを参照してください。

ASP.NET Web アプリケーション サーバーはページをリテラルとサーバー コントロールの階層構造で表します。リテラル コントロールはマークアップ要素 (たとえば HTML タグなど) を表します。サーバー コントロールは出力を動的に生成する ASP.NET クラスを表します。各コンポーネントはページで異なるコンポーネントをレンダリングする役割を果たします。ASP.NET はリテラル コントロールを組み立て、応答ストリームを書き込むサーバー コントロールを起動することによって、HTTP 要求に対する応答を組み立てます。サーバー コントロールは出力を動的に生成し、ページ イベントに応答することができます。

Sitecore のレイアウト エンジンは ASP.NET によって提供されている機能以上の機能を提供します。それらは次のようなものです:

- レイアウト エンジンは .NET ロジックに加えて、XSL 変換を起動することを容易にします。
- レイアウト エンジンを使うと、ブラウザ ベースのユーザー インターフェースを通して、コントロールをプレースホルダーに動的にかつ宣言的にバインドすることが可能になります。
- レイアウト エンジンを使うと、それぞれのプレゼンテーション コンポーネントの出力を異なる基準によってキャッシュすることが可能になります。
- レイアウト エンジンを使った条件付きレンダリングによって、ページ要求に対応するコントロールに対して実行時にロジックを使って作用することが可能になります。

レイアウト エンジンはコンテキスト アイテムのレイアウト詳細のコンテキスト デバイスに指定されたプレゼンテーション コンポーネントを使って HTTP 要求に対応します。またはそのアイテムがレイアウト詳細を定義していない場合は、コンテキスト アイテムに関連付けられたデータ テンプレートのスタンダード バリューのコンテキスト デバイスに指定されたプレゼンテーション コンポーネントを使います。Sitecore はコンテキスト デバイスのレイアウト詳細で指定されたプレゼンテーション コンポーネントの階層構造からページの応答を組み立てます。各プレゼンテーション コンポーネントは Sitecore API を含む任意の .NET API を起動することができます。 また任意のコンテンツ、メタデータ、アイテムの関係、Sitecore のリポジトリーを含む構成設定にアクセスすることができます。

各プレゼンテーション コンポーネントはユーザーのコンテキストに応じて異なる出力を生成することができます。たとえば、ユーザーの認証の可否、ユーザーのプロファイル、セキュリティ認証、要求された言語などに応じた対応が可能です。他の ASP.NET コントロールと同様、プレゼンテーション コンポーネントはイベントに応答することができます。たとえば、ユーザーがコンポーネントをクリックしたとき、などに応答することが可能です。

プレゼンテーション コンポーネントの登録や取り扱いのために、Sitecore はブラウザ ベースの デベロッパー センター アプリケーションを提供しています。開発者は通常、Microsoft Visual Studio およびソース コード管理システムを使って、プレゼンテーション コンポーネントを管理します。

重要

多くの Web ソリューションでは HTTP 要求はディスク上のファイルに対応します。Sitecore の HTTP 要求は

Sitecore データベースのアイテムに対応します。各アイテムはレイアウト詳細を持っています。レイアウト詳細が Sitecore にどのプレゼンテーション コンポーネントを適用するのかを指示します。開発者はプレゼンテーション コンポーネントをディスク上のファイルとして管理します。 たとえば、ASP.NET Web フォーム (`.aspx`) 、XSL 変換ファイル (`.xslt`) 、.NET アセンブリ (`.dll`) などです。レイアウト詳細に関する詳細は、「レイアウト詳細」のセクションを参照してください。

重要

プレゼンテーション コンポーネントは Sitecore の 1 つの定義アイテムと 0 個または複数のディスク上のファイルを含みます。プレゼンテーション コンポーネントを Sitecore のあるインスタンスから別のインスタンスにコピーする場合には、更新されたすべての定義アイテムとファイルを含めてください。また可能な場合には、コード ファイルよりもアセンブリを含めてください。

2.2 レイアウト (ASP.NET .aspx Web フォーム)

レイアウトはできるだけ多くのページで共通な、最も外側の再利用可能なマークアップの構造を定義します。開発者はレイアウトを使用することで:

- 同じマークアップの構造を多くのページ ビューに再利用し、要求されたアイテムのレイアウト詳細に定義に従って、他のプレゼンテーション コンポーネントを内包するプレースホルダーに動的にバインドします。
- 各ページまたは機能 (たとえば、サイトの他のすべてのページと異なるホームページのレイアウトなど) の最も外側のマークアップ層を定義します。
- 異なるタイプのデバイス (たとえば、Web ブラウザ、プリンター、モバイル デバイス、など) からの HTTP 要求に対応して、コンテンツを異なるようにフォーマットします。

2.2.1 レイアウトの実装

Sitecore のレイアウトとは、再利用可能な ASP.NET Web フォーム (`.aspx`) で Sitecore に登録されたものです。ASP.NET では Web フォームを使って HTTP 要求に対応することができます。Sitecore では要求されたアイテムに関連付けられたレイアウトの 1 つを起動して HTTP 要求に対応します。Sitecore は要求と要求されたアイテムのプロパティを使ってどのレイアウトを起動するかを決定します。

レイアウトには Sitecore の定義アイテムとファイル システム上の `.aspx` ファイルが含まれます。レイアウトには C# コード (`.cs`) などの補助的なファイルを含むものもあります。プロジェクトにはコードファイルを .NET アセンブリ (`.dll`) にコンパイルするものもあります。

メモ

Sitecore は C# に加え、ASP.NET のサポートするすべての言語をサポートします。

メモ

開発者は通常は、Sitecore でなくソース コード管理システムを使用して、またパブリッシュでなくリリース管理技術を使用して、ファイル資産を管理します。

Sitecore はレイアウト定義アイテムをコンテンツ ツリーの `/Sitecore/Layout/Layouts` アイテムの下で管理します。レイアウト定義アイテムは `System/Layout/Layout` データ テンプレートを使用します。各レイアウト定義アイテムの [**データ**] セクションの [**Path**] フィールドは `.aspx` ファイルへのパス (Web サイトのドキュメント ルートへの相対パス) を参照します。

Microsoft Visual Studio または デベロッパー センター を使用して設計時に、レイアウトが通常使用するコントロールをレイアウトに静的にバインドします。レイアウト エンジンがそのレイアウトを使って HTTP 要求に対応するたびに、レイアウト ファイル (`.aspx`) の各コントロールをそのコントロールが起動して生成されたマークアップで置換します。

レイアウト エンジンはさらにレイアウトの中で動的にプレースホルダー コントロールにバインドするサーバー コントロールを決定します。レイアウト エンジンはレイアウト ファイル (.aspx) の中の各プレースホルダーを指定されたコントロールの出力で置換します。プレースホルダーに関する詳細は、「プレースホルダー」のセクションを参照してください。レイアウト詳細に関する詳細は、「レイアウト詳細」のセクションを参照してください。

メモ

Sitecore ソリューションはすべての HTTP 要求を処理するためにレイアウトを使うわけではありません。IIS Web サーバーがディスクから静的なファイルを使って対応する HTTP 要求もあります。[2] メディアまたはレイアウト詳細やレイアウトを使用しない他の ASP.NET ハンドラーを起動する要求もあります。`web.config` の `/configuration/sitecore/settings/setting` 要素の name `IgnoreUrlPrefixes` の `value` 属性を構成して、Sitecore が特定の要求を処理せず、ASP.NET がその要求を Sitecore なしで処理するようにすることもできます。

2.2.2 レイアウトの使用

各 HTTP 要求は多くても 1 つのレイアウトを起動します。各論理 Web サイトは通常はサポートされるデバイスごとに少なくとも 1 つのレイアウトを持ちます。たとえば、Web ブラウザとモバイル デバイスをサポートする Web サイトは 1 つのレイアウトを Web ブラウザに、別のレイアウトをモバイル デバイスに使用します。一般に、レイアウトは最も再利用可能度の高いプレゼンテーション コンポーネントです。Sitecore のソリューションでは 1 つのレイアウトを 1 つのデバイスのすべてのページに使用する場合もあります。

メモ

レイアウトはロジックがページ レベルで適用される場合のみコードを持ちます。グローバルなロジックはグローバル アプリケーション (`global.asax`)、要求処理パイプライン、イベント ハンドラーなどの別の機能を使います。アプリケーション ロジックはサブレイアウトなどの個々のプレゼンテーション コンポーネントを使います。サブレイアウトに関する詳細は、「レンダリングとしてのサブレイアウト」のセクションを参照してください。

ヒント

必要なレイアウトの数を最小化するには、プレースホルダーとレイアウト詳細を使ってプレゼンテーション コンポーネントを動的にバインドします。

[2] IIS を構成して ASP.NET を使って要求を処理することに関する詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Dynamic%20Links.aspx から『動的リンク』を参照してください。

2.3 サブレイアウト (ASP.NET .ascx Web ユーザー コントロール)

サブレイアウトはレイアウトで使われる、または他のサブレイアウトでネストされる、マークアップ構造を定義します。開発者はサブレイアウトを使用することで:

- コンテンツをページにレンダリングします。
- 外部システムからデータを読み込みます。
- 複数のページで共有されるマークアップのサブ構造を含め、そのサブ構造がさらにネストされたコンポーネントの上部構造として機能するようにします。
- レイアウトで定義されていて、別のサブレイアウトでは別の特質で定義されている、マークアップ上部構造を再利用します。
- サブレイアウトのプレースホルダーを使用して、コンポーネントを動的に深くネストします。
- ASP.NET アプリケーションをページ全体でなくページのコンポーネントとして実装します。
- 複数のレイアウトにバインドされている再利用可能なコントロールのグループを含めます。

2.3.1 サブレイアウトの実装

サブレイアウトとは、Sitecore に登録された ASP.NET の Web ユーザーコントロールです。レイアウトを起動する各 HTTP 要求は 0 個または複数のサブレイアウトおよび他のプレゼンテーション コンポーネントを起動し、レイアウトを生成することができます。サブレイアウトを静的にレイアウトまたは他のサブレイアウトにバインドすることができます。または動的にレイアウトのプレースホルダーまたはネストされたサブレイアウトにレイアウト詳細を使ってバインドすることができます。レイアウト詳細に関する詳細は、「レイアウト詳細」のセクションを参照してください。

サブレイアウトには Sitecore の定義アイテムとファイル システム上の `.ascx` ファイルが含まれます。サブレイアウトには C# コード (`.cs`) などの補助的なファイルを含むものもあります。プロジェクトにはコードファイルを .NET アセンブリ (`.dll`) にコンパイルするものもあります。

Sitecore はサブレイアウト定義アイテムをコンテンツ ツリーの `/Sitecore/Layout/Sublayouts` アイテムの下で管理します。レイアウト定義アイテムは `System/Layout/Sublayout` データ テンプレートを使用します。各レイアウト定義アイテムの [**データ**] セクションの [**Path**] フィールドは `.ascx` ファイルへのパス (Web サイトのドキュメント ルートへの相対パス) を参照します。

各サブレイアウトはリテラルと動的なサーバー コントロールの階層構造を表すマークアップを含みます。開発者はレイアウト詳細に従ってサーバー コントロールを、サブレイアウトに静的にバインドしたり、サブレイアウトのプレースホルダー コントロールに動的にバインドしたりします。プレースホルダーに関する詳細は、「プレースホルダー」のセクションを参照してください。レイアウト詳細に関する詳細は、「レイアウト詳細」のセクションを参照してください。

2.3.2 サブレイアウトの使用

サブレイアウトは最も柔軟なプレゼンテーション コンポーネントです。サブレイアウトを使用して、小さなマークアップをレイアウトや他のサブレイアウトに埋め込むことができます。Web アプリケーション (フォーム) を実装することができます。再利用可能なコントロールのグループを作成することができます。マークアップを動的に生成することができます。マークアップを生成しない API を起動することができます。これらをはじめ、他の広い用途に使用できます。

メモ

サブレイアウトはレイアウトよりも多いコードファイルを含みます。

2.4 レンダリング

レンダリングとは、パブリッシュされた Web サイトの構成要素として機能する、Sitecore に登録されたプレゼンテーション コンポーネントです。開発者はレンダリングを使用することで:

- コンテンツをページにレンダリングします。
- 外部システムからデータを読み込みます。
- 表示コンポーネントのない、バックエンドのロジックを実行します。たとえば、Web アナリティクスのためのログ要求などです。

2.4.1 レンダリングの実装

レイアウト エンジンはレイアウト以外のすべてのプレゼンテーション コンポーネントに ASP.NET サーバー コントロールを使用します。各レンダリングはレンダリング定義アイテムを含みます。レンダリング定義アイテムはそのレンダリングを実装するコードを含むファイル (レンダリングに渡すパラメーターを含む) を指定します。
Sitecore はいくつかのタイプのサーバー コントロールをサポートします。

- サブレイアウト
- XSL レンダリング
- Web コントロール
- プレースホルダー
- URL レンダリング
- メソッド レンダリング

プレースホルダーは出力をレンダリングしませんが、他のタイプのプレゼンテーション コンポーネントの動的なバインドをサポートします。サブレイアウトに関する詳細は、「サブレイアウト (ASP.NET .ascx Web ユーザー コントロール) 」のセクションを参照してください。他のタイプのレンダリングに関する詳細は「レンダリングの使用」のセクションを参照してください。

メモ

メソッド レンダリングと URL レンダリングはパラメーターと Sitecore が導入した .NET アセンブリ (`.dll` ファイル) を含むレンダリング定義アイテムを含みます。Web コントロールはレンダリング定義アイテムと Web コントロール クラスを含む .NET アセンブリ (`.dll` ファイル) を含みます。XSL レンダリングはレンダリング定義アイテム、変換を起動する Web コントロールを含む Sitecore が導入した .NET アセンブリ、変換コードを含む `.xslt` ファイルを含みます。サブレイアウトはサブレイアウト定義アイテム、サブレイアウトを起動する Web コントロールを含む Sitecore が導入した .NET アセンブリ、サブレイアウトを実装する `.ascx` Web ユーザー コントロール ファイルを含みます。サブレイアウトは 1 つまたは複数の `.cs`、`.dll`、その他のファイルを含むことができます。

2.4.2 レンダリングの使用

開発者はレンダリングをレイアウトとサブレイアウトに設計時に静的にバインドし、レイアウト エンジンがレイアウトやサブレイアウトを処理するたびに、それらのレンダリングを起動させることができます。開発者はレイアウト詳細を使ってレンダリングをレイアウトまたはサブレイアウトのプレースホルダーに実行時に動的にバインドすることもできます。レイアウト詳細に関する詳細は、「レイアウト詳細」のセクションを参照してください。

2.4.3 レンダリングの使用

Sitecore は各種のレンダリング技術をサポートします。特定のタスクで使用する技術の詳細については、Chapter 5 の「プレゼンテーション技術の選択」を参照してください。

レンダリングとしてのサブレイアウト

サブレイアウトを使用してページの出力を生成することができます。特に断りのない場合は、**レンダリング** という用語にはサブレイアウトを含みます。

XSL レンダリング

XSL レンダリングは XSL 変換の結果を出力します。[3] XSL 変換のソースは Sitecore データベースの XML 表現です。XSL レンダリングは XSL の `document()` 関数または XSL 拡張を使って外部リソースにアクセスすることができます。

メモ

XSL 変換は Web クライアントでなく Web サーバーで発生します。

Web コントロール レンダリング

ASP.NET Web コントロールは、クラスとして実装され、通常は `System.Web.UI.WebControls.WebControl` を通して .NET フレームワークの `System.Web.UI.Control` から継承される ASP.NET ページ要素です。ASP.NET Web コントロールは出力を動的に生成し、ページイベントに応答します。これは通常は `Render()` を上書きして出力を生成します。

Sitecore の Web コントロール レンダリングは Sitecore Web コントロール ベース クラス `Sitecore.Web.UI.WebControl` から継承する ASP.NET コントロールです。それは `System.Web.UI.WebControls.WebControl` から継承します。Sitecore の Web コントロール レンダリングは `DoRender()` メソッドを上書きして出力を生成します。

`System.Web.UI.Control` から継承される機能に加え、`Sitecore.Web.UI.WebControl` から継承する Web コントロールは下記をサポートします:

- データ ソース アイテムをコントロールに渡す。
- プレースホルダーへの動的なバインド。
- レンダリングされた出力のキャッシュ。
- Sitecore 開発者のためのコンビニエンス メソッドとプロパティ。

メモ

先述の Sitecore Web コントロール機能を使用しない Web コントロールは、`Sitecore.Web.UI.WebControl` から継承するのでなく、.NET システム クラスから直接継承することができます。

ヒント

レイアウト エンジンは `Sitecore.Web.UI.WebControl` から継承する Web コントロールを Sitecore プレースホルダーに動的にバインドすることができます。ASP.NET ベースクラスから継承する既存の Web コントロールをサブレイ

[3] XSL の詳細については、http://www.w3.org/Style/XSL/ を参照してください。

アウトに静的にバインドすることができます。またそのサブレイアウトをプレースホルダーに動的にバインドすることができます。

メソッド レンダリング

ASP.NET がメソッド レンダリングを処理するときには、パラメータを受け付けない .NET メソッドを起動し、メソッドが出力ストリームに返す文字列を書き出します。

URL レンダリング

ASP.NET が URL レンダリングを処理するときには、URL を起動し、応答を出力ストリームに書き出します。応答が HTML の `<body>` 要素を含む場合は、レイアウト エンジンはその要素のコンテンツのみを出力し、`<body>` 要素自体または body 要素以外の要素、たとえば `<html>`、`<head>`、`<form>` などは出力しません。

メモ

Web クライアントが URL を要求する HTML の `<iframe>` 要素とは異なり、URL レンダリングに指定された URL をサーバーが要求します。

Web パーツ レンダリング

オプションの Web パーツ フレームワークを使うと、Web パーツをレンダリングとして使うことができます。[4]

2.4.4 レンダリング プロパティ

レンダリング プロパティは Sitecore が各レンダリングをどのように使用するかを制御します。レンダリング定義アイテムはデフォルトのレンダリング プロパティを含みます。

注意

デベロッパー センター または **グリッド デザイナー** を使用してレンダリングをレイアウトまたはサブレイアウトに追加する場合、Sitecore はプロパティを定義アイテムからレイアウトまたはサブレイアウト ファイルのレンダリングへの新しい参照にコピーします。レンダリング定義アイテムのレンダリング プロパティを変更した場合、Sitecore はそのレンダリングを参照しているすべてのレイアウトとサブレイアウトの対応するパラメーターを更新しません。

メモ

開発者はレンダリング プロパティを制御します。ユーザーはレンダリング パラメーターを使ってレンダリングを制御することができます。レンダリング パラメーターに関する詳細は、「パラメーター テンプレート レンダリング パラメーター」のセクションを参照してください。

ヒント

CMS ユーザーがレンダリング パラメーターを含めデフォルトのレンダリング プロパティの別のグループを選択できるようにするには、別のレンダリング プロパティとパラメーターを提供する複数のレンダリング定義アイテムを 1 つのレンダリング コードに挿入することでできます。

[4] Sitecore での Web パーツの使用の詳細については、http://sdn.sitecore.net/Resources/Free%20Modules/Web%20Part%20Framework.aspx から『Sitecore Web パーツ フレームワーク』を参照してください。

説明 レンダリング プロパティ

ユーザーが**ページ エディター**のデザイン モードで、マウスを使ってレンダリングの上をホバーしたとき、Sitecore はレンダリング定義アイテムの [**エディター オプション**] セクションの [**説明**] プロパティを表示します。[5]

レンダリング設定データ テンプレートと パラメーター テンプレート レンダリング プロパティ

レンダリング設定データ テンプレートを使って [**コントロール プロパティ**] ダイアログを定義し、CMS ユーザーにレンダリング パラメーターを入力するためのカスタム フォームを提供することができます。

Sitecore はレンダリング定義アイテムの [**エディター オプション**] セクションの [**パラメーター テンプレート**] レンダリング プロパティで指定されたデータ テンプレートを使って、ユーザーがレンダリング パラメーターを定義するときに [**コントロール プロパティ**] で表示されるフィールドを定義することができます。レンダリング パラメーターに関する詳細は、「レンダリング パラメーター」のセクションを参照してください。[6]

レンダリング設定データ テンプレートは 1 つまたは複数のデータ テンプレート セクションを含む構造体を定義します。各セクションは 1 つまたは複数のデータ テンプレート フィールドを含みます。[7] [**コントロール プロパティ**] ダイアログはレンダリング設定データ テンプレートに基づきデータ エントリー インターフェースを表示します。

レンダリング設定データ テンプレートは `System/Layout/Rendering Parameters/Standard Rendering Parameters` データ テンプレートから継承します。`System/Layout/Rendering Parameters/Standard Rendering Parameters` データ テンプレートはすべてのタイプのレンダリングに共通するレンダリング パラメーターを定義します。レンダリングにレンダリング パラメーター テンプレートを指定しなかった場合、Sitecore は `System/Layout/Rendering Parameters/Standard Rendering Parameters` データ テンプレートをそのデータ テンプレートのパラメーター テンプレートとして使用します。

レンダリング設定データ テンプレートを使用してアイテムを作成することはできません。Sitecore は**コントロール プロパティ** ダイアログを、現行アイテムに関連付けられたデータ テンプレートに関連付けられたレンダリング設定データ テンプレートに基づいて生成します。そのアイテムのレイアウト詳細に入力されたデータを保存します。レイアウト詳細に関する詳細は、「レイアウト詳細」のセクションを参照してください。

注意

レンダリング設定データテンプレートを更新したとき、たとえばフィールドの名前を変更または削除した場合、Sitecore はレイアウト詳細を自動的には更新しません。(そのフィールドを名前で参照している場合がありえます)

重要

レンダリング設定データ テンプレートはスタンダード バリューをサポートしません。それに代えて、**パラメーター** というレンダリング プロパティを使用します。**パラメーター** という名前のレンダリング プロパティに関する詳細は、「パラメーター レンダリング プロパティ」のセクションを参照してください。

---

[5] **ページ エディター**についての詳細は、
http://sdn.sitecore.net/Reference/References%20in%20Japanese/Client%20Configuration%20Cookbook.aspx から『クライアント構成クックブック』を参照してください。

[6] **コントロール プロパティ**についての詳細は、
http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20Cookbook.aspx から『プレゼンテーション コンポーネント クックブック』を参照してください。

[7] **データ テンプレート**についての詳細は、
http://sdn.sitecore.net/Reference/References%20in%20Japanese/Data%20Definition%20Reference.aspx から『データ定義リファレンス マニュアル』を参照してください。

追加後にプロパティを開く レンダリング プロパティ

レンダリング定義アイテムの [**エディター オプション**] セクションの [**追加後にプロパティを開く**] レンダリング プロパティの値が [**はい**] の場合、ユーザーがレンダリングをレイアウト詳細に追加したあとで Sitecore は [**コントロール プロパティ**] ユーザー インターフェースを表示します。[8] レイアウト詳細に関する詳細は、「レイアウト詳細」のセクションを参照してください。

追加後にプロパティを開く レンダリング プロパティは [**レンダリングの選択**] ダイアログの「このダイアログを閉じた後でプロパティを開く」のチェックボックスに影響します。

次のテーブルは [**追加後にプロパティを開く**] レンダリング プロパティの値による影響を説明しています。

Open Properties After Add	このダイアログを閉じた後でプロパティを開く
Default	有効で、チェックなし
No	無効で、チェックなし
Yes	有効で、チェックあり

カスタマイズ ページ レンダリング プロパティ

レンダリング定義アイテムの [**エディター オプション**] の [**カスタマイズ ページ**] プロパティはレガシーな仕様で、レンダリング設定データ テンプレートで置換されるものです。

注意

[**カスタマイズ ページ**] レンダリング プロパティでなく、[**パラメーター テンプレート**] レンダリング プロパティを使用してください。カスタマイズ ページ機能を実装している場合、パラメーター テンプレートへの移行を検討してください。

プレースホルダー レンダリング プロパティ

レンダリング定義アイテムの [**データ**] セクションの [**プレースホルダー**] レンダリング プロパティはレンダリングのデフォルトのプレースホルダー キーを指定します。ユーザーがレイアウト詳細でプレースホルダー キーを指定していない場合、または開発者が条件付きレンダリングを使って実行時にレンダリングをレイアウト詳細に追加する場合にプレースホルダー キーを指定していない場合、Sitecore は **プレースホルダー** レンダリング プロパティをプレースホルダー キーとして使用します。条件付きレンダリングに関する詳細は、「条件付きレンダリング」のセクションを参照してください。

メモ

条件付きレンダリングを使って、レンダリングに関連付けられているプレースホルダーを実行時に変更することができます。

パラメーター レンダリング プロパティ

レンダリング定義アイテムの [**データ**] セクションの [**パラメーター**] レンダリング プロパティはレンダリング パラメーターのデフォルト値を指定します。レンダリング パラメーターに関する詳細は、「レンダリング パラメーター」のセクションを参照してください。

[8] **コントロール プロパティ**についての詳細は、
http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20Cookbook.aspx から『プレゼンテーション コンポーネント クックブック』を参照してください。

メモ

URL クエリ文字列 パラメーター エンコードを使って、レンダリング定義アイテムの [**パラメーター**] レンダリング プロパティを定義します。[**パラメーター**] レンダリング プロパティに保存する値を決定するには、一時的なレイアウトまたはサブレイアウトにレンダリングを静的にバインドし、次に **デベロッパー センター**または**グリッド デザイナー**を使ってレンダリングのプロパティを定義し、さらにレイアウトまたはサブレイアウトのソースを参照し、コントロールの **パラメーター** 属性の値をレンダリング定義の [**パラメーター**] フィールドにコピーします。

キャッシュ レンダリング プロパティ

キャッシュ レンダリング プロパティはレンダリングのデフォルト出力のキャッシュ オプションを提供します。キャッシュ オプションに関する詳細は、Chapter 4 の「出力のキャッシュ」を参照してください。

Sitecore は、デベロッパー センターまたはグリッド デザイナーを使ってレンダリングをレイアウトまたはサブレイアウトに静的にバインドしたときに、キャッシュ オプションをレンダリング定義アイテムからコントロールにコピーします。レイアウト詳細を使ってレンダリングをプレースホルダーに動的にバインドしたときに、レイアウト詳細でキャッシュ オプションを指定しない限り、レイアウト エンジンはレンダリング定義アイテムから動的にキャッシュ オプションを適用します。

メモ

条件付きレンダリングを使ってキャッシュ レンダリング プロパティを動的に設定することができます。条件付きレンダリングに関する詳細は、「条件付きレンダリング」のセクションを参照してください。

特定のタイプのレンダリング プロパティ

このセクションにおけるレンダリング プロパティについての一般的な記述に加え、いくつかのタイプのレンダリングではプロパティを使うことができます。[9]

2.4.5 レンダリング パラメーター

レンダリング パラメーターはレンダリングの動作を制御します。開発者はレンダリング パラメーターを使用することで:

- コンテンツや構成、その他のレンダリングのデータのハードコーディングを避けることができます。
- レンダリングを別の構成で再利用できます。
- ユーザーがレンダリングの機能を制御できるようにします。

レンダリング パラメーターを 3 つの場所で適用することができます: レンダリング定義アイテム、レンダリングを静的にバインドした場合のレイアウトまたはサブレイアウト、さらにレンダリングをプレースホルダーに動的にバインドした場合のレイアウト詳細、の 3 つです。

メモ

条件付きレンダリングを使って、レンダリング パラメーターを実行時に変更することができます。条件付きレンダリングに関する詳細は、「条件付きレンダリング」のセクションを参照してください。

重要

Microsoft Visual Studio を使ってレンダリングをレイアウトまたはサブレイアウトに静的にバインドする場合は、レンダリ

[9] 特定のタイプのレンダリングのレンダリング プロパティに関する詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20Cookbook.aspx から『プレゼンテーション コンポーネント クックブック』を参照してください。

ング パラメーターをレイアウトまたはサブレイアウトで定義します。レンダリング定義アイテムが対応するプロパティとパラメーターを定義している場合でも同様です。

ヒント
レンダリング プロパティをレンダリング定義アイテムで定義して Sitecore のユーザー インターフェースを制御します。レンダリングを静的にレイアウトまたはサブレイアウトにバインドしレンダリングの機能を制御する場合、またレイアウト詳細を使ってレンダリングを動的にプレースホルダーにバインドする場合には、レンダリング パラメーターを定義します。デフォルトのレンダリング パラメーターを「パラメーター レンダリング プロパティ」のセクションで記載されているように定義します。

プレースホルダー レンダリング パラメーター

プレースホルダー レンダリング パラメーターはプレースホルダーのデフォルト キーを指定します。
レイアウト詳細を使ってレンダリングをプレースホルダーに動的にバインドする場合にはプレースホルダーのキーを指定する必要があります。レイアウトまたはレイアウト エンジンが処理したサブレイアウトのいずれかの中に、レンダリングを処理する前にプレースホルダーが存在する場合、かつユーザーがそのレンダリングに読み取りアクセス権をもつ場合、かつ条件付きレンダリングのルールがレンダリングの実行されることを妨げない場合、Sitecore は指定されたキーを使ってレンダリングを動的にプレースホルダーにバインドします。条件付きレンダリングに関する詳細は、「条件付きレンダリング」のセクションを参照してください。

メモ
レンダリング プロパティ ダイアログを含めた Sitecore のユーザー インターフェースの中で、[**プレースホルダー**] はレンダリングをプレースホルダーにバインドする時にのみ意味があります。

データ ソース レンダリング パラメーター

各レンダリングはそれから処理を始める 1 つのデータ ソース アイテムを指定することができます。開発者はレンダリング データ ソースを使用することで:

- アイテム パスまたは CSS クラスや ID などの GUID をハードコーディングしないようにします。
- レンダリングを別のデータを使って再利用します。
- レンダリングが処理するデータを指定します。

データ ソースをレンダリングに渡さない場合は、コンテキスト アイテムがレンダリングのデフォルトのデータ ソースです。コンテキスト アイテムに関する詳細は、「コンテキスト アイテム」のセクションを参照してください。
レイアウト詳細を使ってレンダリングをプレースホルダーに動的にバインドする場合、またはレンダリングをレイアウトまたはサブレイアウトに静的にバインドする場合には、データ ソースを指定することができます。

キャッシュ レンダリング パラメーター

デフォルトのキャッシュ オプションを提供するキャッシュ レンダリング プロパティに加え、レイアウト詳細を使ってレンダリングを動的にプレースホルダーにバインドする場合またはレンダリングを静的にバインドする場合は、キャッシュ レンダリング パラメーターを定義することができます。出力のキャッシュに関する詳細は、Chapter 4 の「出力のキャッシュ」を参照してください。

メモ
レイアウト、サブレイアウト、レイアウト詳細で定義されたキャッシュ レンダリング パラメーターはレンダリング定義アイテム

で定義されたキャッシュ レンダリング プロパティを上書きします。キャッシュ レンダリング プロパティに関する詳細は、「キャッシュ レンダリング プロパティ」のセクションを参照してください。

パーソナライゼーション レンダリング パラメーター

パーソナライゼーション レンダリング パラメーターを使うとユーザーはレンダリングに適用する条件付きレンダリング ルールを選択することができます。条件付きレンダリングに関する詳細は、「条件付きレンダリング」のセクションを参照してください。

テスト レンダリング パラメーター

テスト レンダリング パラメーターはレンダリングが起動されたときに適用する多変量解析を指定します。[10]

追加パラメーター レンダリング パラメーター

追加パラメーター レンダリング パラメーターを使うとユーザーはレンダリング パラメーターをキーと対応する値のリストとして入力することができます。レンダリング定義アイテムで指定されたパラメーター テンプレートに対応するフィールドがないパラメーターに対して、**追加パラメーター** レンダリング パラメーターを使うことができます。パラメーター テンプレートに関する詳細は、「レンダリング設定データ テンプレートと パラメーター テンプレート レンダリング プロパティ」のセクションを参照してください。

パラメーター テンプレート レンダリング パラメーター

デフォルトの**レンダリング** パラメーターに加え、**パラメーター テンプレート**を使ってカスタムのレンダリング パラメーターを定義することができます。Sitecore はレンダリングの**コントロール プロパティ** ユーザー インターフェースの**パラメーター テンプレート**で定義されたフィールドを表示します。

XSL レンダリングでは、カスタム レンダリング パラメーターは、ルート `<xsl:stylesheet>` 要素の中の `<xsl:param>` 要素を使って作成されたパラメーターに対応します。.NET レンダリングでは、カスタム レンダリング パラメーターはレンダリング定義アイテムで指定された .NET オブジェクトのプロパティに対応します。

[10] 多変量解析に関する詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/OMS-Analytics%20Configuration%20Reference.aspx から『OMS アナリティクス構成リファレンス』を参照してください。

2.4.6 FieldRenderer Web コントロール

Sitecore は単一のフィールド値を出力する FieldRenderer Web コントロールを提供します。これは自動的に動的リンクを展開し、ページ エディターでインライン編集コントロールを提供します。[11] 開発者は FieldRenderer Web コントロールを使って単一のフィールド値を読み込み、フォーマットします。

FieldRenderer Web コントロールは `DataSource` パラメーターを公開します。これは Sitecore がフィールド値を読み込むアイテムを制御します。`DataSource` パラメーターを指定しない場合は、レイアウト エンジンはコンテキスト アイテムからフィールドを処理します。コンテキスト アイテムに関する詳細は、「コンテキスト アイテム」のセクションを参照してください。

FieldRenderer Web コントロールは次のパラメーターもサポートします:

パラメーター	機能
`After`	フィールド値の後に出力するテキスト (ある場合には、閉じる `EnclosingTag` の中で)
`Before`	フィールド値の前に出力するテキスト (ある場合には、開く `EnclosingTag` の中で)
`DisableWebEditing`	フィールドのインライン編集を無効にするためには True
`EnclosingTag`	フィールド値を囲むマークアップ要素 (たとえば、`div`)
`FieldName`	処理するフィールドの名前

`Sitecore.Kernel` アセンブリの `Sitecore.Web.UI.WebControls.FieldRenderer` クラスが FieldRenderer Web コントロールの実装を提供します。

`/Sitecore/Layout/Renderings/System/FieldRenderer` Web コントロール レンダリング定義アイテムがこのクラスを参照し、このコントロールをデベロッパー センターのレイアウト エンジンまたはサブレイアウトにドラッグすることを容易にします。

[11] 動的リンクに関する詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Dynamic%20Links.aspx から『動的リンク』を参照してください。

2.5 プレースホルダー

Sitecore のプレースホルダーは、他のコントロールがレイアウト詳細に従って実行時に動的にバインドするレイアウトとサブレイアウトによる名前のついた領域を定義する ASP.NET コントロールです。[12] 開発者はプレースホルダーを使用することで:

- その中で異なるコンポーネントが異なる要求を起動する、再利用可能なレイアウトまたはサブレイアウトの領域を表します。
- マークアップ構造体の異なる領域で異なるプレゼンテーション コンポーネントを、そのマークアップ構造体と重複することなく、起動します。

2.5.1 プレースホルダーの実装

プレースホルダーはレイアウトまたはサブレイアウトの中での位置をプレースホルダー キーという名前で関連付けます。レイアウト エンジンはプレースホルダーを、要求されたアイテムのレイアウト詳細のキーで関連付けられたサブレイアウトとレンダリングで、動的に置換します。一貫性がユーザビリティーにつながり、ユーザビリティーが再訪ビジターにつながります。開発者は一貫性をコンテンツとコードの再利用を通して実現します。プレースホルダーは一貫性を最大化し、プレゼンテーション コンポーネントの再利用を通じて開発と保守を最小化します。これは多くのタイプのコンテンツにわたり、別の論理サイトに対しても可能です。

Sitecore のプレースホルダーは `Key` をはじめとするプロパティを公開する Web コントロールです。他の Web コントロールと異なり、プレースホルダー Web コントロールはレイアウトまたはサブレイアウトに常に静的にバインドします。レイアウト詳細を使ってプレースホルダーをプレースホルダーに動的にバインドすることはありません。

各レイアウトとサブレイアウトはプレースホルダーをいくつでも含むことができます。プレースホルダーはネストをいくつでもサポートします。サブレイアウトは、レイアウトの中のプレースホルダーにバインドしているサブレイアウトの中のプレースホルダーにバインドすることができます。レイアウト詳細を使って任意の数のプレゼンテーション コンポーネントを各プレースホルダーにバインドすることができます。レイアウト詳細が同じプレースホルダー キーを使って複数のプレゼンテーション コンポーネントを関連付けた場合、レイアウト エンジンはそれらのコンポーネントをプレースホルダーにレイアウト詳細で指定された順序でバインドします。

Sitecore がインストールした .NET アセンブリ (`.dll` ファイル) のクラスはプレースホルダー Web コントロールを含みます。コンテンツ ツリーにプレースホルダー定義アイテムを挿入する必要はありません。プレースホルダー設定定義アイテムを挿入することによって、ユーザーがプレースホルダーにバインドできるレンダリングを制御することができます。[13]

メモ

`web.config` の `/configuration/sitecore/settings/setting` 要素の name `LayoutPageEvent` の `value` が、どの ASP.NET ページイベントによってレイアウト エンジンがレイアウト詳細を適用するかを制御します。開発者は `PreInit` イベント、`Init` イベント、または `Load` イベントの間に、プレゼンテーシ

[12] Sitecore のプレースホルダーを ASP.NET のマスター ページのコンテンツ プレースホルダーと混同しないようにしてください。特に断りのない限り、Sitecore の文書でプレースホルダーという用語は Sitecore のプレースホルダーを意味します。

[13] プレースホルダー設定についての詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Client%20Configuration%20Cookbook.aspx から『クライアント構成クックブック』を参照してください。

ョン コンポーネントをプレースホルダーにバインドすることを選択することもできます。

2.5.2 プレースホルダー キー

各プレースホルダーはテキスト キーを持ちます。レイアウト詳細に指定された各プレゼンテーション コンポーネントはプレースホルダーのキーを示します。これらのレイアウト詳細の中のプレースホルダー キーの参照がレイアウト エンジンに対して、ページ ビューの生成時にどのプレゼンテーション コンポーネントを動的に各プレースホルダーにバインドするかを、またコンポーネントを各プレースホルダーにバインドする順序を、指定します。

プレースホルダーキーはあるアイテムのデバイスのレイアウト詳細で参照されるすべてのプレゼンテーション コンポーネントの中でユニークである必要があります。あるアイテムのプレゼンテーション詳細の 1 つのデバイスに使われるレイアウトとサブレイアウトが、1 つの共通のキーを持つ複数のプレースホルダーを含むのは無効です。たとえば、プレースホルダーを含み、レイアウトのプレースホルダーと同じキーを使ったネストしたサブレイアウトなどです。1 つのページで一緒に使われることのない複数のコンポーネントで定義された共通の領域のプレースホルダーは、1 つのプレースホルダー キーを共用することができます。たとえば、1 つのページビューで一緒に使われることのない 2 つのサブレイアウトは、共通のキーをもつプレースホルダーをそれぞれに含むことが可能です。

レイアウト詳細はキーまたは完全修飾キーによりプレースホルダーを参照することができます。完全修飾プレースホルダー キーは、コンポーネントのネストした階層構造の中でのプレースホルダーの場所を示します。たとえば、キー `B` を持つプレースホルダーを含むサブレイアウトが、レイアウトの中でキー `A` をもつプレースホルダーにバインドしたとき、ネストしたプレースホルダーの完全修飾キーは `/A/B` です。ページ エディターのデザイン モードは常に完全修飾プレースホルダーキーを使いますが、ユーザーはレイアウト詳細では非完全修飾キーを入力することができます。

2.5.3 プレースホルダーの設定

プレースホルダー設定はプレースホルダーのプロパティを提供します。たとえば、ユーザーがどのサブレイアウトとレンダリングを各プレースホルダーにバインドできるかを制御することなどです。[14]

2.5.4 プレースホルダーの使用

開発者はプレースホルダーを使って異なる条件の下で異なるプレゼンテーション コンポーネントを含む再利用可能なレイアウトとサブレイアウトの領域を表します。たとえば、同じレイアウトを使って複数のアイテムを処理する場合などです。開発者はレイアウト詳細を使ってレイアウト エンジンに、異なるプレゼンテーション コンポーネントをプレースホルダーにバインドさせることができます。

各レンダリング コンポーネントは出力を動的に生成することができます。プレースホルダーを使うと、共通のレイアウトを共有する異なるアイテムに対して異なるレンダリング コンポーネントを実行することができます。

ヒント

レイアウト詳細の管理を最小化するために、プレースホルダーは必要な場合のみに使用します。レンダリングをレイアウト詳細のプレースホルダーに動的にバインドする代わりに、可能な限りプレゼンテーション コンポーネントをレイアウトまたはサブレイアウトに静的にバインドします。

[14] プレースホルダー設定についての詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Client%20Configuration%20Cookbook.aspx から『クライアント構成クックブック』を参照してください。

2.6 Sitecore ユーザー インターフェース プレゼンテーション コンポーネント

Sitecore は XML レイアウト、XML コントロール、XML ダイアログ、XML フォームを使って、CMS のユーザー インターフェースを実装しています。開発者はパブリッシュされる Web サイトの開発のためには通常はこれらを使用せず、CMS のユーザー インターフェースのコンポーネントに使われているのみであるため、この文書ではこれらの技術は説明しません。

Chapter 3

要求の処理

この章では Sitecore が ASP.NET のページ ライフサイクルに付加するレイヤー機能を説明します。

この章ではまず、Sitecore のアイテムへの要求を処理する Sitecore のレイアウト エンジンについて説明します。次にデバイスとレイアウト詳細の概念を説明します。これによってレイアウト エンジンは発生した HTTP 要求のプロパティに応じて、異なるプレゼンテーション コントロールをアイテムに適用できます。

この章には次のセクションがあります。

- Sitecore のレイアウト エンジン
- デバイス
- レイアウト詳細

3.1 Sitecore のレイアウト エンジン

Sitecore のレイアウト エンジンは ASP.NET のページ ライフサイクルを次の方法によって拡張しています:

- HTTP モジュールが、ユーザー、言語、要求されたアイテム、デバイス、その他のコンテキスト情報を示すコンテキスト オブジェクトを定義する。
- Web フォーム (`.aspx` ファイル) で指定されたコンポーネントからページをアセンブルするのでなく、レイアウト詳細で指定されたコンポーネントからレイアウト エンジンがページをアセンブルする。

Web アプリケーションは各種の Web クライアントからの HTTP 要求に応答します。たとえば、ブラウザ、RSS リーダー、モバイル、その他のデバイスなどです。Sitecore のレイアウト エンジンは各 HTTP 要求のプロパティを使って、どんなタイプのデバイスがアイテムを要求しているのか、どのように応答を組み立てればよいか、を判別します。レイアウト エンジンはコンテキスト アイテムのレイアウト詳細のコンテキスト デバイスに指定されたプレゼンテーション コンポーネントを起動し、HTTP 応答を組み立てます。レイアウト詳細に関する詳細は、「レイアウト詳細」のセクションを参照してください。コンテキスト アイテムに関する詳細は、「コンテキスト アイテム」のセクションを参照してください。
URL をファイル システムのファイルでなくデータベースのアイテムにマッピングすることにより、Sitecore は異なるプレゼンテーション コンポーネントを動的に起動し、異なる条件の下で 1 つのアイテムへの要求に対応します。たとえば、異なるタイプのデバイスがアイテムを要求したときに異なるようにコンテンツをフォーマットすることなどです。またレイアウト エンジンは条件付きレンダリング、アナリティクス、その他の機能を提供します。[15] 条件付きレンダリングに関する詳細は、「条件付きレンダリング」のセクションを参照してください。

3.1.1 コンテキスト アイテム

コンテキスト アイテムは要求のライフサイクルにおいて、デフォルトのデータ ソースです。レイアウト エンジンはコンテキスト アイテムを要求された URL のパスに基づいて判別します。
コンテキスト アイテムはページの要求に関連した多くの動作のデフォルトのデータ ソース アイテムです。たとえば、適用するレイアウト詳細の判別などです。コンテキスト アイテムはデータ ソースを指定していないレンダリングのデフォルトのデータ ソースです。レンダリングのデータ ソースに関する詳細は、「データ ソース レンダリング パラメーター」のセクションを参照してください。
たとえば、デフォルトの構成では、Web クライアントがパス `/hr/jobs.aspx` を要求した場合、レイアウト エンジンはコンテキスト アイテムをコンテキスト データベースの`/Sitecore/Content/Home/hr/jobs` アイテムに設定します。

[15] アナリティクスに関する詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/OMS-Analytics%20Configuration%20Reference.aspx から『OMS アナリティクス構成リファレンス』を参照してください。

3.2 デバイス

デバイスとは、インターネットに接続され、他の接続されたデバイス (たとえば、Web サーバー) に対して HTTP 要求を発する、さまざまなタイプの Web クライアントを表します。各デバイスは異なるタイプの Web クライアントを表します。各デバイスはそれぞれのマークアップの要件を持つことができます。レイアウト エンジンはコンテキスト アイテムのレイアウト詳細のコンテキスト デバイスに指定されたプレゼンテーション コンポーネントを適用します。
開発者はデバイスを使って、各種の Web クライアントに対して異なるプレゼンテーション コンポーネントの集合を使用してコンテキスト アイテムをフォーマットします。コンテキスト アイテムに関する詳細は、「コンテキスト アイテム」のセクションを参照してください。

3.2.1 デバイスの実装

レイアウト エンジンは下記の方法で各 HTTP 要求に対してコンテキスト デバイスを判別します:

- カスタムの .NET ロジックを適用する。
- URL クエリ文字列パラメーターを登録されたデバイスと比較する。
- Web クライアントのユーザーエージェントを登録されたデバイスと比較する。
- コンテキスト サイトのプロパティから判別する。

HTTP 要求が特定のデバイスを起動しない場合、レイアウト エンジンは **デフォルト** デバイスをコンテキスト デバイスと想定します。**デフォルト** デバイスは通常は Web ブラウザを表します。

フォールバック デバイス

各デバイスにフォールバック デバイスを関連付けることができます。コンテキスト アイテムがコンテキスト デバイスのためのレイアウト詳細を含んでいない場合は、レイアウト エンジンはフォールバック デバイスのレイアウト詳細を適用します。コンテキスト アイテムに関する詳細は、「コンテキスト アイテム」のセクションを参照してください。

メモ

もし定義されている場合、コンテキスト アイテムのフォールバック デバイスのレイアウト詳細は、そのアイテムに関連付けられたデータ テンプレートのスタンダード バリューのコンテキスト デバイスのレイアウト詳細を上書きします。もしコンテキスト アイテムがコンテキスト デバイスのレイアウト詳細を含んでいないが、フォールバック デバイスのレイアウト詳細は含んでいる場合は、レイアウト エンジンはアイテムに関連付けられたデータ テンプレートのスタンダード バリューのコンテキスト デバイスのレイアウト詳細を評価せずに、フォールバック デバイスのコンテキスト アイテムのレイアウト詳細を適用します。

3.2.2 デバイスの使用

Sitecore はデフォルトで 2 つのデバイスを提供します。

- **デフォルト** デバイスは通常は Web ブラウザを表します。
- **印刷**デバイスはプリンターを表します。

デフォルトでは、URL クエリ文字列がパラメーター `p` に値 `1` (`p=1`) を含んでいる場合、レイアウト エンジンは**印刷**デバイスを起動します。レイアウト エンジンが**印刷**デバイスを起動しない場合は、**デフォルト** デバイスを起動します。
さらに追加の Web クライアントのタイプを表すカスタムデバイスをいくつでも登録することができます。またレイアウト エンジンが別の条件の下で他のプレゼンテーション コンポーネントを使う必要のある場合にもそうすることができます。追加の

デバイスには下記のようなものがありますが、この限りではありません:

- RSS リーダー
- モバイル デバイス
- 特定の Web ブラウザ
- XML、たとえば Microsoft Silverlight や Adobe の Flash コンポーネントが使用するもの
- 1 つのコンテンツアイテムの複数の表現、たとえば別のマーケティング ブランドなど

重要

各カスタム デバイスを起動する基準を定義します。

3.3 レイアウト詳細

レイアウト詳細は、レイアウト エンジンが起動し各種のデバイスからのアイテムの要求に対応するための、再利用可能なプレゼンテーション コンポーネントを指定します。レイアウト詳細を使用することで:

- レイアウト エンジンが起動して各種のアイテムの HTTP 要求に対応するための、レイアウト、サブレイアウト、レンダリングを制御します。
- プレゼンテーション コンポーネントを宣言的に再利用し複数のアイテムの要求に対応します。
- 1 つのアイテムの別のタイプの Web クライアントに対する複数のプレゼンテーションを定義します。

3.3.1 レイアウト詳細の実装

コンテキスト アイテムのレイアウト詳細は、異なるタイプのデバイスがアイテムを要求したときに起動するレイアウトとレンダリングを指定します。デバイスに関する詳細は、「デバイス」のセクションを参照してください。

各アイテムの各デバイスのレイアウト詳細はどのレイアウトが適用されるかを指定します。またどのサブレイアウトとレンダリングがレイアウトとネストしたサブレイアウトで各プレースホルダーを生成するかを指定します。レイアウト詳細はレイアウト エンジンに、共通のレイアウトとサブレイアウトと同じプレースホルダーを、異なるアイテムまたは異なるタイプのアイテムには異なるコンポーネントで、動的に生成するように指示します。各プレゼンテーション コンポーネントは Sitecore データベースのデータ、Sitecore API、その他の API または ASP.NET や XSL で利用できる他のリソースを使って、出力を動的に生成することができます。

レイアウト詳細は実行時までデータをプレゼンテーションから分離し、コンテンツとプレゼンテーション コンポーネントの再利用、管理の柔軟性、サイト全体にわたるユーザー インターフェースの変更 (たとえばリブランドなど) 、サブサイトにおける別のプレゼンテーション、その他のユーザー インターフェースの管理要件への対応をもたらします。

多くの他のデータ テンプレートに継承されている標準テンプレートは、レイアウト詳細を含むフィールドを定義します。[16]

ヒント

管理を最小化するためには、各コンテンツ アイテムでレイアウト詳細を定義するより、各データ テンプレートのスタンダード バリューでレイアウト詳細を定義します。[17]

3.3.2 レイアウト デルタ

レイアウト デルタを使用するとアイテムがクローン アイテムまたはデータ テンプレートのスタンダード バリューからレイアウト詳細の一部を継承することができます。

レイアウト詳細を継承しているアイテムのレイアウト詳細を更新すると、レイアウト詳細を含むフィールドは継承しているレイアウト詳細との差分のみを保存します。ベース アイテムのレイアウト詳細の変更はレイアウト デルタを含むアイテムに動的に適用されます。

レイアウト デルタは累積します。クローンのレイアウト デルタはクローン元のアイテムのレイアウト詳細に適用され、それはスタンダード バリューに適用されます。

---

[16] 標準テンプレートについての詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Data%20Definition%20Reference.aspx から『データ定義リファレンス マニュアル』を参照してください。

[17] スタンダード バリューについての詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Data%20Definition%20Reference.aspx から『データ定義リファレンス マニュアル』を参照してください。

Sitecore CMS 6.3 およびそれ以前のバージョン

アイテム テンプレートのスタンダード バリュー アイテムでレイアウト詳細を定義します。
レイアウト詳細を変更する手順:

- 開発者の場合: コンテンツ エディターを使用してテンプレートのスタンダード アイテムのレイアウト詳細を編集します。または個々のコンテンツ アイテムのレイアウト詳細をカスタマイズします。
- ビジネス ユーザーの場合: ページ エディターのデザイン モードを使用します。標準レイアウトとして保存 のオプションはこれらの値をテンプレートのスタンダード バリュー アイテムに保存することを意味します。変更したレイアウトを現在のアイテムのみに保存することも可能です。

開発者がテンプレート スタンダード アイテムのレイアウトを変更する場合、その変更はそのテンプレートに基づくすべてのアイテムに継承されます。
しかし、カスタムのレイアウト詳細をもつアイテムはスタンダード バリュー アイテムから値を継承せず、テンプレートのスタンダード バリュー アイテムのレイアウト変更に影響を受けません。

Sitecore CMS 6.4 およびそれ以降のバージョン

アイテム テンプレートのスタンダード バリュー アイテムでレイアウト詳細を定義します。
レイアウト詳細を変更する手順:

- 開発者の場合: コンテンツ エディターを使用してテンプレートのスタンダード アイテムのレイアウト詳細を編集します。または個々のコンテンツ アイテムのレイアウト詳細をカスタマイズします。
- ビジネス ユーザーの場合: ページ エディターのデザイン モードを使用します。レイアウト詳細の変更を保存すると、変更は現在のアイテムのみに保存され、アイテムのテンプレート スタンダード バリューには保存されません。ビジネスユーザーはリセット ボタンをクリックすることによって、テンプレート スタンダード バリューにあるアイテムのレイアウト詳細をリセットすることが可能です。開発者にはさらに多くの選択肢があります。

開発者がテンプレート スタンダード バリューのレイアウトを変更する場合、その変更はそのテンプレートに基づくすべてのアイテムとクローンに継承されます。
カスタムのレイアウト詳細をもつアイテムは、テンプレートの新しいレイアウト詳細とユーザーが行ったレイアウトのカスタマイズの両者を併せて使用してレンダリングされます。この両者の相違点がレイアウト デルタです。

3.3.3 レイアウト詳細 vs. ASP.NET のコンテンツ とマスター ページ

ASP.NET はマスター ページをサポートしますが、マスター ページはレイアウトの機能のいくつかを提供します。マスター ページ (`.aspx` 拡張子を使います) はコンテンツとプレゼンテーションを制御する参照マスター ページ (`.master` ファイル) を含みます。[18]ASP.NET マスター ページは ASP.NET が複数のコンテンツ ページに適用するコントロールを含みます。
Sitecore はファイル システムでなくデータベースに保存される抽象コンテンツ ストレージと宣言的レイアウト詳細を提供し、ASP.NET のページ生成プロセスを拡張します。レイアウト ファイルはマスター ページを参照する ASP.NET のコンテンツ ページであることが可能です。Sitecore の開発者は通常はマスターページとコンテンツ ページを避けます。これは宣言的なレイアウト詳細による、より高い柔軟性と再利用性のためです。

[18] ASP.NET マスター ページとコンテンツ ページに関する詳細は、http://msdn2.microsoft.com/en-us/library/wtxbf3hh.aspx を参照してください。

Sitecore のアイテムは論理的には ASP.NET のコンテンツ ページと類似しています。その中にはコンテンツと参照のプレゼンテーション コンポーネントを含んでいますが、ASP.NET のコンテンツ ページが提供するよりはるかに大きな柔軟性を持ちます。1 つのマスター ページを参照するより、コンテンツ アイテムは異なるタイプのデバイスがアイテムを要求したときに起動する異なるレイアウトとレンダリングを参照することができます。ASP.NET のコンテンツ ページと異なり、アイテムを任意の数の言語に翻訳することができ、アイテムを任意の数のデバイスに対して異なるレイアウトとレンダリングを使って処理することができます。

Sitecore のレイアウトは、いくつかのコンテンツ アイテムに適用できるコンテンツを含んでいるという点で、ASP.NET のマスター ページと類似しています。マスター ページが一階層のネストのみ可能であるのに対して、レイアウト詳細はプレゼンテーション コンポーネントの宣言的なネストを何階層でもサポートします。

ASP.NET のマスター ページはページを元にした体系を提供しているので、ナビゲーションの開発、コンテンツの再利用、その他の動的な機能の実装などにおいて困難が生じる場合があります。Sitecore のレイアウト エンジンはデータ中心の体系を提供し、ナビゲーション、再利用、その他の動的な機能を容易にします。

3.3.4 条件付きレンダリング

条件付きレンダリングを使ってレンダリングを動的に表示したり非表示にしたり、またレンダリング プロパティとパラメーターを設定したりすることができます。[19] **コントロール プロパティ** ダイアログの、レンダリングの、**パーソナライゼーション** レンダリング パラメーターで、条件付きレンダリング ルールを選択します。**パーソナライゼーション** レンダリング パラメーターに関する詳細は、「パーソナライゼーション レンダリング パラメーター」のセクションを参照してください。

[19] 条件付きレンダリングに関する詳細は、http://sdn.sitecore.net/Reference/Sitecore%206/Rules%20Engine%20Cookbook.aspx から『ルール エンジン クックブック』を参照してください。

3.4 プレゼンテーション コンポーネント定義アイテム

すべてのプレゼンテーション コンポーネントはディスク上のファイルを起動します。たとえば、コンパイルされた .NET アセンブリのクラス、または `.ascx`、`.aspx`、`.xslt`、その他のコード ファイルなどです。プレゼンテーション コンポーネントは CSS、JavaScript、メディア、その他のディスク上のファイルに依存することがよくあります。それらは Sitecore で対応する定義アイテムがあることもないこともあります。

レイアウト、サブレイアウト、XSL レンダリング、その他のタイプのアイテムは **Path** という名前のフィールドを含む定義アイテムから構成されます。これはプレゼンテーション ロジックを実装しているディスク上のファイルへのパスを指定しています。Web コントロール レンダリング定義アイテムとメソッド レンダリング定義アイテムのフィールドはレイアウト エンジンが起動する .NET コンポーネントを判別するための情報を含んでいます。

注意

定義アイテムを移動、複製、名前の変更、削除した場合は、Sitecore は **Path** フィールドを自動的に更新しません。ファイルを作成、移動、複製、名前の変更、削除した場合、Sitecore は自動的に対応する定義アイテムを作成、移動、複製、名前の変更、削除はしません。

重要

プレゼンテーション コンポーネントをある環境から別の環境にコピーする場合、新規と更新のファイルと対応する定義アイテムを含めてください。

Chapter 4

出力のキャッシュ

この章では各プレゼンテーション コンポーネントの出力をキャッシュし、パフォーマンスとスループットを最大化するために使うことのできる、レイアウト エンジンの機能を説明します。

この章ではまず各レンダリングをキャッシュする際の各種の基準を説明し、次に Sitecore がレンダリングした出力をそれらの基準によってどのようにキャッシュするのかを説明します。

この章には次のセクションがあります。

- レンダリングされた出力のキャッシュのオプション
- レンダリングされた出力のキャッシュの実装

4.1 レンダリングされた出力のキャッシュのオプション

各プレゼンテーション コンポーネントは異なる条件の下で異なる出力を生成する場合があります。たとえば、フッターのレンダリングはすべてのページで同じ出力を生成するが、ブレッドクラムのレンダリングは異なるアイテムで異なる出力を生成し、ナビゲーションのレンダリングは異なるユーザーにコンテンツ アイテムへのアクセス権に応じて異なる出力を生成します。

レイアウト エンジンは各レンダリングの出力をキャッシュすることができます。開発者はレンダリングされた出力のキャッシュを使うことにより、コンポーネントを再度起動する代わりに、そのコンポーネントによって同じ条件の下で以前に生成された出力を読み込むことにより、パフォーマンスを改善することができます。出力のキャッシュはデータの構造とコードの最適化の必要性を低減するわけではありませんが、特に大規模なソリューションにおいては、キャッシュはパフォーマンスを大きく向上させることがあります。

重要

キャッシュはソリューション全体のパフォーマンスにとって非常に重要です。Sitecore のソリューションのスループットとキャパシティを向上させるための最も効率的な方法は、出力のキャッシュの構成を頻繁に最適化することです。

重要

ASP.NET のページイベントに応答するコンポーネントの出力を、影響に対する十分な理解なしにキャッシュしないでください。

重要

Sitecore のレンダリングした出力のキャッシュと、Web フォームと Web ユーザー コントロールの `OutputCache` ディレクティブで実装される ASP.NET ページとフラグメントのキャッシュを混同しないでください。開発者は ASP.NET ページとフラグメントのキャッシュを Sitecore のコンテンツで使用するべきではありません。また必要な場合には ASP.NET のキャッシュをクリアする必要があります。 (たとえば、Sitecore パブリッシュ操作の後など) Sitecore の文書では、キャッシュという用語は特に断りのない場合は Sitecore のレンダリングした出力のキャッシュを意味します。

4.2 レンダリングされた出力のキャッシュの実装

デフォルトではレイアウト エンジンは各 HTTP 要求に対して出力のキャッシュなしに、各プレゼンテーション コンポーネントを実行します。

重要

プレゼンテーション コンポーネントを使用するたびに、キャッシュの基準を選択する必要があります。

メモ

カスタマイゼーションなしではメソッド レンダリングと URL レンダリングの出力をキャッシュすることはできません。

レイアウト エンジンは各管理 Web サイトに対して個別の出力キャッシュを管理します。キャッシュは管理 Web サイトごとに自動的に変わります。

各出力キャッシュは任意の数のキーと値の組のリストから構成されます。各キャッシュ キーはプレゼンテーション コンポーネントと各種のキャッシュの基準を表すユニークな文字列です。そのキャッシュ キーに対応するキャッシュの値はその基準の下でそのコンポーネントの出力です。1 つのプレゼンテーション コンポーネントの複数の起動は異なるキャッシュ キーでキャッシュを生成し、異なる条件の下で 1 つのレンダリングが生成する出力を表します。

各キャッシュ キーは自動的にコンテキスト言語を含みます。キャッシュ キーはレンダリング定義アイテムの ID またはサブレイアウト ファイルや XSL レンダリング ファイルへのパスなどの、プレゼンテーション コンポーネントの識別子を含みます。キャッシュはコンポーネントごと、また言語ごとに自動的に変わります。

重要

Web コントロールの出力をキャッシュするには、`GetCachingID()` メソッドを上書きし、レンダリングの識別子を返します。たとえば、レンダリング定義アイテムまたは名前空間とクラス名の GUID などです。

メモ

キャッシュは複数の基準によって変わります。たとえば、プレゼンテーション コンポーネントのキャッシュをデータ ソースとユーザーによって変化するように構成することができます。

次のセクションで説明する *VaryBy* プロパティを True に設定すると、Sitecore はキャッシュ キーに対応するトークンを追加し、キャッシュはそれらのプロパティによって変化することになります。出力をキャッシュするように構成されたプレゼンテーション コンポーネントをレイアウト エンジンが評価するとき、対応するキーをもつエントリーがキャッシュに存在する場合、それはキャッシュされた出力を読み出します。

プレゼンテーション コンポーネントにキャッシュが構成されていないか、キャッシュが対応するエントリーを含んでいない場合は、レイアウト エンジンはコンポーネントを起動します。コンポーネントにキャッシュ プロパティがある場合、レイアウト エンジンは対応するキーをもつエントリーをキャッシュに追加します。

重要

メモリの使用とシステムが各コンポーネントを起動する回数を最小化するため、可能な限り少ない基準で各サブレイアウトとレンダリングの出力をキャッシュします。

メモ

デフォルトではパブリッシュは出力キャッシュをクリアします。

4.2.1 どちらのキャッシュ設定が適用されるか？

Sitecore では開発者は出力キャッシュの基準を 3 つの場所で定義することができます。

- サブレイアウトとレンダリング定義アイテムの [**キャッシュ**] セクション。
- プレゼンテーション コンポーネントのプロパティ (レイアウトまたはサブレイアウトに静的にバインドする場合)
- **コントロール プロパティ** ダイアログの [**キャッシュ**] セクション (プレゼンテーション コンポーネントをレイアウト詳細のプレースホルダーにバインドする場合)

レイアウト エンジンは、2 つの条件の下で、定義アイテムの [**キャッシュ**] セクションで定義されたキャッシュ基準を使います。

- 開発者が**デベロッパー センター**または**グリッド デザイナー**を使ってレイアウトまたはサブレイアウトにレンダリングをバインドした場合、Sitecore はキャッシュ プロパティをレンダリング定義アイテムからコントロールにコピーします。(プレゼンテーション コンポーネントへの静的な参照)
- レイアウト詳細が動的にプレースホルダーにバインドされたプレゼンテーション コンポーネントのキャッシュ基準を指定しない場合。

レイアウト詳細を使ってレンダリングをプレースホルダーに動的にバインドした場合、レイアウト詳細で明示的に定義されたキャッシュ設定はレンダリング定義アイテムで定義されたキャッシュ設定を上書きします。定義アイテムで定義されたキャッシュ設定は、**コントロール プロパティ** ダイアログの [**キャッシュ**] セクションにキャッシュ設定が存在しない場合のみ適用されます。

4.2.2 出力のキャッシュのプロパティ

キャッシュ可能なプレゼンテーション コンポーネントは次のキャッシュ プロパティをサポートします。すべてのキャッシュ プロパティはデフォルトでは False になっています。

Cacheable

各プレゼンテーション コンポーネントの **Cacheable** プロパティはレイアウト エンジンがそのコンポーネントの出力をキャッシュするかどうかを制御します。**Cacheable** プロパティが False の場合、レイアウト エンジンはコンポーネントへの参照を処理するたびに、コンポーネントを起動します。次のセクションで定義されるいずれの *VaryBy* プロパティにかかわらず、レイアウト エンジンはコンポーネントの出力をキャッシュせず、キャッシュからの読み出しもしません。

Cacheable プロパティが True で、どの *VaryBy* プロパティも True でない場合、レイアウト エンジンはプレゼンテーション コンポーネントを、管理 Web サイトで各言語でその最初の使用の際に起動し、その出力をキャッシュに書き込み、キャッシュされた出力を続くそのコンポーネントのその管理サイトと言語に対するすべての使用の際に読み出します。

Cacheable プロパティが True で 1 つまたは複数の *VaryBy* プロパティが **True** である場合、それらの *VaryBy* プロパティがレイアウト エンジンがコンポーネントを起動するか、以前に同じ *VaryBy* 条件の下で生成されキャッシュされた出力を読み出すかを決定します。

次のように **Cacheable** 属性を設定します:

- False として、どのレンダリングにもレイアウト エンジンが出力をキャッシュしないようにする。
- True として、どのレンダリングにもレイアウト エンジンが出力をキャッシュするようにする。
- True とし、論理サイトと言語以外のどの基準でも出力が変わらないようにするコンポーネントには *VaryBy* プロパティを True にしない。
- True とし、 指定した条件の下では異なる出力を生成する 1 つまたは複数のコンポーネントには *VaryBy* を

True とする。

メモ
プレゼンテーション コンポーネントの **Cacheable** プロパティが False の場合、*VaryBy* プロパティは効力を持ちません。レイアウト エンジンは **Cacheable** プロパティが False または定義されていないコンポーネントの出力をキャッシュしません。

データにより変更

データにより変更 プロパティはプレゼンテーション コンポーネントのデータ ソースに基づいて出力のキャッシュが変わるかどうかを制御します。
次のように**データにより変更** プロパティを設定します:

- 別のデータ ソースと使われるとき別の出力を生成しないコンポーネントには False とする。
- 別のデータ ソースと使われるとき別の出力を生成するコンポーネントには True とする。

デバイスにより変更

デバイスにより変更 プロパティはコンテキスト デバイスの名前に基づいてキャッシュが変わるかどうかを制御します。
次のように**デバイスにより変更** プロパティを設定します:

- 別のデバイスと使われるとき別の出力を生成しないコンポーネントには False とする。
- 別のデバイスと使われるとき別の出力を生成するコンポーネントには True とする。

ログインにより変更

ログインにより変更 プロパティはユーザーが認証されているかどうかに基づいて出力のキャッシュが変わるかどうかを制御します。
次のように**ログインにより変更** プロパティを設定します:

- 認証ビジターに対して非認証ビジターと別の出力を生成しないコンポーネントには False とする。
- 認証ビジターに対して非認証ビジターと別の出力を生成するコンポーネントには True とする。

メモ
ログインにより変更 プロパティを起動するキャッシュ構成について、レイアウト エンジンはすべての匿名ユーザーを単一の認証ユーザーとして扱います。

パラメーターにより変更

データにより変更 プロパティはプレゼンテーション コンポーネントに渡されたレンダリング パラメーターに基づいて出力のキャッシュが変わるかどうかを制御します。
開発者は次のように**パラメーターにより変更** プロパティを設定します:

- 別のレンダリング パラメーターが渡されたとき別の出力を生成しないコンポーネントには False とする。
- 別のレンダリング パラメーターが渡されたとき別の出力を生成するコンポーネントには True とする。

メモ
Sitecore の以前のバージョンで構築されたソリューションはトークン **パラメーターにより変更** でなく **VaryByParam** を使用しているかもしれません。**VaryByParam** を使っているインスタンスは **パラメーターにより変更** に更新してください。

クエリ文字列により変更

クエリ文字列により変更 プロパティは URL で渡されるクエリ文字列パラメーターに基づいて出力のキャッシュが変わるかどうかを制御します。

開発者は次のように**クエリ文字列により変更** プロパティを設定します:

- 別のクエリ文字列パラメーターが提供されたとき別の出力を生成するコンポーネントには True とする。
- 別のクエリ文字列パラメーターが提供されたとき別の出力を生成しないコンポーネントには False とする。

メモ

パラメーターにより変更 プロパティは開発者によって渡されるレンダリング パラメーターの値に基づいて出力のキャッシュを変化させます。**クエリ文字列により変更** プロパティは URL クエリ文字列で渡されるパラメーターの値に基づいて出力のキャッシュを変化させます。

ユーザーにより変更

ユーザーにより変更 プロパティはコンテキスト ユーザーのドメインとユーザー名によって出力のキャッシュが変わるかどうかを制御します。

開発者は次のように**ユーザーにより変更** プロパティを設定します:

- 別のユーザーに別の出力を生成しないコンポーネントには False とする。
- 別のユーザーに別の出力を作成するコンポーネントには True とする。ただしパブリッシュ操作の間のアクティブユーザーの数が比較的少ない場合。

メモ

ユーザーにより変更 はすべての匿名ユーザーを単一の認証ユーザーとして扱います。

メモ

過度のメモリ消費を避けるため、比較的少ないユーザー数のソリューションの場合やキャッシュの使用率を注意深く監視する場合を除き、**ユーザーにより変更** プロパティの使用は避けてください。

メモ

ログインにより変更 プロパティは、匿名ユーザーと認証ユーザーを区別し、ユーザーが認証されているかどうかによって、Sitecore に異なる出力を生成させます。**ユーザーにより変更** プロパティは Sitecore に各ユーザーごとに異なる出力を生成させます。

Chapter 5

プレゼンテーション技術の選択

この章では利用できるプレゼンテーション コンポーネント技術を選択するためのガイダンスを提供します。

まず、一般的なプレゼンテーション技術の考慮点を説明し、続いて XSL レンダリング、サブレイアウト、Web コントロールに特有な考慮点を論じます。

この章には次のセクションがあります。

- 一般的なプレゼンテーション技術の考慮点
- 特定のプレゼンテーション技術の考慮点

5.1 一般的なプレゼンテーション技術の考慮点

各プレゼンテーション コンポーネントの技術を選択する場合には次の点を考慮します。

- レイアウトは多くのページ ビューで共有されるマークアップの上部構造を表します。
- プレースホルダーはレイアウト エンジンがレイアウト詳細に従って動的にレンダリングをバインドするレイアウトとサブレイアウトの領域を表します。
- 既存の Web フォームと Web ユーザーコントロールは容易にサブレイアウトに変換できます。
- レイアウトとサブレイアウトのみがプレースホルダーを含むことができます。
- Web コントロール レンダリングは容易に既存の Web コントロールを包みこむかまたは置換できます。
- メソッド レンダリングは容易に既存の .NET メソッドを包みこむことができます。
- Web パーツ レンダリングは容易に既存の Web パーツを参照できます。
- URL レンダリングは他の URL から読み込まれたコンテンツを埋め込みます。
- XML レイアウト、XML コントロール、XML ダイアログ、XML フォームは CMS ユーザー インターフェースとして適しています。

上記の考慮点によってもプレゼンテーション コンポーネント技術の選択が明確でない場合には、サブレイアウト、または Web コントロール レンダリング、または XSL レンダリングのいずれかとしてコンポーネントを実装することを検討してください。

メモ

各ページでは複数の技術を実装したプレゼンテーション コンポーネントを含むことが可能です。

一般に、XSL レンダリングは Sitecore のアイテムの階層構造を処理してフィールド値を読み取りマークアップを生成するコンポーネントに適し、サブレイアウトと Web コントロール レンダリングはかなりのロジックを含むプレゼンテーション コンポーネントに適しています。

5.2 特定のプレゼンテーション技術の考慮点

次のセクションではプレゼンテーション コンポーネントを実装するために使用できる特定の技術のメリットとデメリットを説明します。

5.2.1 サブレイアウトの考慮点

サブレイアウトは ASP.NET Web ユーザー コントロールのすべての機能を提供します。サブレイアウトはデザイン (`.ascx` ファイル) とロジック (オプションのコードビハインドまたはコードビサイド ファイル) を分離します。サブレイアウトは Sitecore のコンテキスト、Sitecore のデータベース、.NET API、ASP.NET のすべてのリソースと API にアクセスできます。サブレイアウトは Sitecore プレースホルダーを含む、ネストした ASP.NET コントロールをサポートします。開発者はコンパイルされたサブレイアウトのコードを Visual Studio のデバッガーを使ってステップ実行できます。

5.2.2 XSL レンダリングの考慮点

XSL と XPath 構文の少々の知識があれば、XSL はマークアップを生成するための強力な言語になります。XSL は HTML に似ていますが、ロジックを使って出力を動的に生成します。XSL は W3C で定義されたオープンな標準です。XSL は、この構文に慣れた技術者にとっては、各種のプレゼンテーションのタスクにとって効率的で美しい手段です。

XSL は特に XML を HTML または XHTML に変形するようなマークアップの生成に非常に適しています。XSL は通常、フィールド値を読み出して整形するのに適しています。またサイトマップやブレッドクラムなどの再帰的な機能にも適しています。XSL はとくにプロトタイピングには有効です。開発者は必要な場合には XSL レンダリングを .NET 技術の 1 つに変換することができます。

XSL はデベロッパー センターまたは Microsoft Visual Studio で編集可能なテキストファイルを使用します。XSL レンダリングを更新した場合、ASP.NET は再起動しません。一方、コンパイルされた .NET レンダリングを編集した場合には ASP.NET は再起動します。

Sitecore の XSL 拡張コントロールと関数は自動的に CMS データベースのフィールド値のインライン編集をサポートします。

デベロッパー センターの編集ウィンドウの下のプレビュー フレームを使うと、開発者はデータ ソース アイテムを選択して コードを編集しながら XSL レンダリングの出力をリアルタイムで参照することができます。

XSL と XPath の構文、および XSL コードで使用できる共通プログラミング機能の欠如のために、XSL は複雑なロジックには不向きです。宣言的なプログラミング モデルは多くの開発者にとって馴染みの少ないものでしょう。XSL コードの複雑な XPath とその他のステートメントは保守にとっても困難です。一般に XSL は複数のデータ ソース、特に XML 以外のリソースを使う場合には不向きです。

開発者は通常 XSL を Microsoft Visual Studio などの統合開発環境で編集しますが、オフラインの XSL エディターは Sitecore データベースの XML 表現にアクセスすることはできず、 .NET XSL 拡張を起動することはできません。開発者は Visual Studio を使用して XSL レンダリングをデバッグすることはできませんが、XSL レンダリングは Sitecore デバッガーで参照できるトレースを書き出すことができます。XSL レンダリングはコンパイル時のエラー検出機能を提供せず、実行時の例外管理のみです。

XSL レンダリングはソース コードを実行するので、それを本番環境を含むすべての環境に展開する必要があります。動的に解釈される XSL はネイティブの .NET コードのパフォーマンスに及びません。

プロジェクトでの XSL の使用の可否によらず、Sitecore の開発者は ASP.NET を理解することが必要です。XSL で可能なことはすべて.NET で可能であるので、XSL は開発者の実装とサポートのためのスキルとしてはオプショナルな技

術です。XSL と .NET の両方を使用することは類似なロジックを 2 つの言語で実装することになります。
XSL レンダリングは XSL 拡張を通して .NET ロジックを起動することができ、コンパイルされた .NET コードによるロジックを使った柔軟な XSL マークアップによるプレゼンテーション コントロールが可能となります。XSL 変換のパフォーマンスはネイティブの .NET コンポーネントには及びませんが、XSL のフォーマット機能のメリットはパフォーマンスの差を補います。これは特にキャッシュできる出力を生成するコンポーネントで当てはまります。
XSL レンダリングはネストしたプレースホルダーまたは ASP.NET コントロールを含むことはできません。

5.2.3 Web コントロールの考慮点

Web コントロールのメリットはサブレイアウトのメリットに似ています。サブレイアウトのメリットに関する詳細は、「サブレイアウトの考慮点」のセクションを参照してください。
Web コントロール レンダリングは ASP.NET Web コントロールのすべての機能をサポートします。Web コントロールは Sitecore のコンテキスト、Sitecore のデータベース、.NET API、.NET のすべてのリソースと API にアクセスできます。これらの機能は XSL レンダリングで利用できる機能のスーパーセットです。開発者はコンパイルされた Web コントロールのコードを Visual Studio のデバッガーを使ってステップ実行できます。
サブレイアウトと異なり、Web コントロールはプレースホルダーを含むことができませんが、コントロールをプログラムにより追加することは可能です。さらに重要なことは、サブレイアウトはデザインを ASP.NET コードファイルを使用したプレゼンテーションから分離しません。Web コントロールは予め定義されたマークアップをあまりまたは全く使用しないで、マークアップを動的に生成するコンポーネントに適します。