



Sitecore CMS 7.0

プレゼンテーション コンポーネント API クックブック

CMS 開発者のためのコンセプトの概要

目次

Chapter 1	イントロダクション.....	5
Chapter 2	Sitecore のコンテキスト.....	6
2.1	Sitecore のコンテキストの概要	7
2.2	コンテキスト アイテム	8
2.3	コンテキスト サイト	9
2.3.1	コンテキスト サイトのホーム アイテム	9
2.4	コンテキスト データベース.....	10
2.5	コンテキスト言語	11
2.6	コンテキスト デバイス.....	12
2.6.1	サンプル コード: コンテキスト デバイスの設定	12
2.7	コンテキスト ドメイン.....	14
2.8	コンテキスト ユーザー	15
2.9	コンテキストの生の URL	16
2.10	コンテキスト ページ モード	17
2.10.1	サンプル コード: メタデータ フィールドの公開	17
Chapter 3	アイテムにアクセスする	18
3.1	アイテムへのアクセスの概要	19
3.1.1	コンテンツ アイテムのわかりやすい URL にアクセスする方法	19
3.1.2	メディア アイテムのわかりやすい URL にアクセスする方法.....	19
3.1.3	アイテムにデバイスのためのレイアウト詳細があるかどうかを知る方法	19
3.2	アイテムの値を使って ASP.NET コントロールを生成する.....	21
3.2.1	アイテムの子を ASP.NET コントロールにバインドする方法	21
	アイテムのリストをコントロールにバインドする方法	21
3.2.2	アイテムをコントロールにバインドする際のフィールド値の使用法	22
3.2.3	アイテムのデータを使って ASP.NET リピーターを生成する	22
3.3	アイテムの子孫にアクセスする方法.....	25
3.3.1	サンプル コード: サイト マップ Web コントロール	25
3.4	アイテムの祖先にアクセスする方法.....	27
3.4.1	サンプル コード: ブレッドクラム Web コントロール.....	27
3.4.2	サンプル コード: 整形されたナビゲーション Web コントロール.....	28
3.5	アイテムを並べ替える.....	29
3.5.1	子の並べ替えのルール	29
3.5.2	子の並べ替えのルールの選択方法	30
3.5.3	カスタムのアイテムの比較機能の実装方法	30

サンプル コード: アイテムをフィールド値で並べ替える	30
3.5.4 カスタムの子の並べ替えのルールの実装方法	30
3.5.5 アイテムのリストを並べ替える方法	31
3.5.6 サンプル コード: 最新のアイテム Web コントロール	32
Chapter 4 フィールドにアクセスする	36
4.1 フィールドへのアクセスの概要	37
4.1.1 RTE フィールドの動的なリンクをわかりやすい URL に変換する方法	37
4.2 renderField パイプライン	38
4.3 FieldRenderer Web コントロール	39
4.4 フィールド タイプ Web コントロール	40
4.4.1 リンク フィールド Web コントロール	40
4.4.2 イメージ フィールド Web コントロール	40
4.4.3 日付フィールド Web コントロール	41
4.4.4 テキスト フィールド Web コントロール	41
4.5 フィールドへのアクセスのサンプル	42
4.5.1 サンプル コード: 選択フィールドのアイテムへのリンク	42
4.5.2 サンプル コード: ファイルまたはイメージ フィールドのメディアへのリンク	42
4.5.3 サンプル コード: 複数選択フィールドのアイテムへのリンク	43
4.5.4 サンプル コード: ファイル ドロップ エリア フィールドのメディア アイテムへのリンク	43
Chapter 5 言語の使用	45
5.1 言語の概要	46
5.2 言語を起動するサンプル コード	47
5.2.1 サンプル コード: コンテキスト言語のバージョン データを持つアイテムへのリンク	47
5.2.2 サンプル コード: 言語国旗 Web コントロール	47
5.2.3 サンプル コード: ユーザー プロファイルからコンテキスト言語を知る	48
言語プロセッサ ベース クラス	49
5.2.4 サンプル コード: Web クライアントの好みから言語を決定する	50
5.2.5 サンプル コード: スレッド カルチャーの設定	52
5.2.6 フォールバック言語のサポート	52
サンプル コード: フィールドのフォールバック言語	54
サンプル コード: アイテムのフォールバック言語	55
Chapter 6 エラー処理	57
6.1 例外処理の概要	58
6.1.1 ASP.NET の例外処理	58
6.2 Try/Catch ブロック	60
6.3 Sitecore のエラーページ	61

6.4	XSL レンダリングの例外の捕捉方法	62
6.4.1	エラー ヘルパー.....	64
6.5	Web コントロール レンダリングの例外の捕捉方法	66
6.6	レイアウトの例外の捕捉方法	67
6.7	アプリケーションの例外の捕捉方法	68

Chapter 1

イントロダクション

この文書ではプレゼンテーション コンポーネントで使用するのことができる API (Application Programming Interfaces) について説明します。Sitecore のレイアウト、サブレイアウト、Web コントロールを開発する前に、また XSL レンダリングのための .NET 拡張機能を実装する前に、この文書をお読みいただくことをお勧めします。

この文書ではまず Sitecore のコンテキストの概要を説明します。Sitecore のコンテキストとは Sitecore が処理する現行の HTTP 要求とコントロールに関する情報を含んでいます。次の 2 つの章では、アイテムとフィールド値にアクセスするために使用する API について説明します。さらに続く章では、各アイテムで複数の言語によって使用できる API について説明します。最後の章では、エラーと例外の処理に関するテクニックを説明します。

この文書には次の章があります。

- Chapter 1 – イントロダクション
- Chapter 2 – Sitecore のコンテキスト
- Chapter 3 – アイテムにアクセスする
- Chapter 4 – フィールドにアクセスする
- Chapter 5 – 言語の使用
- Chapter 6 – エラー処理

Chapter 2

Sitecore のコンテキスト

この章では Sitecore のコンテキストの概要を説明します。まず Sitecore のコンテキストの概要を説明し、次にコンテキスト アイテム、サイト、データベース、言語、デバイス、ドメイン、ユーザー、生の URL、ページモードについて説明します。

この章には次のセクションがあります。

- Sitecore のコンテキストの概要
- コンテキスト アイテム
- コンテキスト サイト
- コンテキスト データベース
- コンテキスト言語
- コンテキスト デバイス
- コンテキスト ドメイン
- コンテキスト ユーザー
- コンテキストの生の URL
- コンテキスト ページ モード

2.1 Sitecore のコンテキストの概要

Sitecore のコンテキストは現行の HTTP 要求に関する情報を含み、レイアウト エンジンを含む Sitecore の機能を制御します。このセクションでは Sitecore のコンテキストを定義する `Sitecore.Context` クラスのプロパティを説明します。

レイアウト エンジンは `web.config` で定義された `httpRequestBegin` パイプラインを起動し、各 HTTP 要求に対する Sitecore のコンテキストを定義します。API を使って Sitecore コンテキストで実行される任意のコードから Sitecore のコンテキスト プロパティを設定することができます。たとえば、.NET プレゼンテーション コンポーネントなどです。`httpRequestBegin` 要求処理パイプラインを追加、削除、置換して、Sitecore がどのようにコンテキストを定義するかをカスタマイズすることができます。

メモ

この文書に記載しているサンプル コードでは、`output` という名前の `System.Web.UI.HtmlTextWriter` オブジェクトの存在を前提としているものもあります。

プレゼンテーション コンポーネントについての詳細は、『[プレゼンテーション コンポーネント リファレンス](#)』と『[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。

ASP.NET ログイン コントロールの使用と、Web サイトのユーザーを認証し、ユーザー プロファイルを管理するための Sitecore API についての詳細は、『[セキュリティ API クックブック](#)』を参照してください。

この文書で説明されている API についての詳細は、『[コンテンツ API クックブック](#)』を参照してください。

さらなる Sitecore API についての情報は、『[Sitecore API リファレンス](#)』を参照してください。

2.2 コンテキスト アイテム

コンテキスト アイテムとは現行の HTTP 要求によって起動された Sitecore のアイテムです。通常はコンテキスト アイテムは現行の HTTP 要求の URL のパスに対応します。`HttpRequest.Begin` パイプラインの中の、アイテムの解決プロセッサ (`Sitecore.Pipelines.HttpRequest.ItemResolver`) がコンテキスト アイテムを設定します。

通常はコンテキスト アイテムは現行 URL のパスに対応します。たとえばデフォルトの構成では、Sitecore はパス `/section/page.aspx` を含む URL を処理するときは、コンテキスト アイテムを `/Sitecore/Content/Home/Section/Page` に設定します。

メモ

コンテキスト アイテムは必ずしも現行 URL のパスに直接対応するわけではありません。たとえば HTTP 要求が別名を起動する場合、URL は別名定義アイテムに対応しますが、コンテキスト アイテムはその別名定義アイテムに参照されたアイテムに対応します。

エイリアスに関する詳細は、『[コンテンツ リファレンス](#)』を参照してください。

`Sitecore.Data.Items.Item` クラスと `Sitecore.Context.Item` プロパティを使ってコンテキスト アイテムにアクセスすることができます。たとえば：

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
```

多くのプレゼンテーション コンポーネントはコンテキスト アイテムに依存します。たとえば、コンテンツ本文レンダリングはコンテキスト アイテムからフィールド値を読み出すことができます。ブレッドクラム レンダリングはコンテキスト アイテムの祖先へのリンクを出力することができます。ナビゲーション レンダリングはコンテキスト アイテムとその祖先へのリンクを整形して情報体系 (Information Architecture) の中でのコンテキスト アイテムの場所を示すことができます。

2.3 コンテキスト サイト

コンテキスト サイトとは現行の HTTP 要求によって起動された、Sitecore に管理された Web サイトです。`web.config` の `/configuration/sitecore/sites/site` 要素が管理 Web サイトのプロパティを定義します。各属性の目的は `web.config` のコメントに記されています。`HttpRequest.BeginPipeline` の中の、サイトの解決プロセッサ

(`Sitecore.Pipelines.HttpRequest.SiteResolver`) が HTTP 要求のプロパティに合致する最初のサイト定義をコンテキスト サイトに設定します。

`Sitecore.Sites.SiteContext` クラスと `Sitecore.Context.Site` プロパティを使ってコンテキスト サイトにアクセスすることができます。たとえば：

```
Sitecore.Sites.SiteContext contextSite = Sitecore.Context.Site;
```

プレゼンテーション コンポーネントは、様々な目的のためにコンテキストのプロパティを使うことができます。

2.3.1 コンテキスト サイトのホーム アイテム

管理 Web サイトのホーム アイテムはそのサイトのホーム ページに対応します。

`Sitecore.Context.Site.StartPath` プロパティを使ってコンテキスト サイトのホーム アイテムにアクセスすることができます。コンテキスト サイトのホーム アイテムを知る方法：

```
Sitecore.Sites.SiteContext contextSite = Sitecore.Context.Site;
Sitecore.Data.Items.Item home =
Sitecore.Context.Database.GetItem(contextSite.StartPath);
```

プレゼンテーション コンポーネントの中にはホーム アイテムに依存するものもあります。たとえば、ロゴ レンダリングではホーム アイテムからイメージを読み出し、ホーム アイテムへのリンクを含めることができます。ナビゲーション レンダリングでは、ホーム アイテムの子のそれぞれへのリンクを出力することができます。

2.4 コンテキスト データベース

コンテキスト データベースとは、現行の HTTP 要求に関連付けられた Sitecore のデータベースです。

`HttpRequest.Begin` パイプラインの中の、データベースの解決プロセッサ

(`Sitecore.Pipelines.HttpRequest.DatabaseResolver`) がコンテキスト データベースを設定します。各管理 Web サイトは、そのサイトのデフォルトのコンテキスト データベースを指定することができます。デフォルトのコンテキスト データベースは、コンテキスト サイトに関連付けられたデータベースです。Sitecore のユーザー インターフェイスと URL のクエリ文字列パラメーターは、コンテキスト サイトに指定されたデフォルトのコンテキスト データベースと異なるコンテキスト データベースを指定することができます。

`Sitecore.Data.Database` クラスと `Sitecore.Context.Database` プロパティを使ってコンテキスト データベースにアクセスすることができます。たとえば：

```
Sitecore.Data.Database contextDatabase = Sitecore.Context.Database;
```

重要

すべての環境で正常に機能するために、プレゼンテーション コンポーネントは常に、データベースを名前参照するのではなく、コンテキスト データベースを使用します。

重要

プレゼンテーション コンポーネントは、Sitecore データベースを更新しません。

2.5 コンテキスト言語

コンテキスト言語とは、現行の HTTP 要求に関連付けられた言語です。httpRequestBegin パイプラインの中の、言語の解決プロセッサー（`Sitecore.Pipelines.HttpRequest.LanguageResolver`）がコンテキスト言語を設定します。特に指定されていない限り、API はコンテキスト言語の各アイテムの現行のバージョンからフィールド値を読み出します。

`Sitecore.Globalization.Language` クラスと `Sitecore.Context.Language` プロパティを使ってコンテキスト言語にアクセスすることができます。たとえば：

```
Sitecore.Globalization.Language contextLanguage = Sitecore.Context.Language;
```

コンテキスト言語の getter プロパティは、次のリストから最初に定義された値を返します：

- プロパティの値
- コンテキスト サイトに関連付けられた Sitecore の言語の cookie の値
- コンテキスト サイト定義の `language` 属性の値
- `web.config` の `/configuration/sitecore/settings/setting` 要素の `name` `DefaultLanguage` である `value` 属性

`sc_lang` クエリ文字列パラメーターが有効な言語を指定している場合、またはリンク マネージャー構成の `languageEmbedding` 属性が `never` 以外の値であり URL の最初のパスのステップが言語を指定している場合、言語の解決プロセッサーはコンテキスト言語を設定します。「サンプル コード: ユーザー プロファイルからコンテキスト言語を知る」と「サンプル コード: Web クライアントの好みから言語を決定する」のセクションを参照してください。

リンク管理構成に関する詳細は、『[コンテンツ API クックブック](#)』を参照してください。

2.6 コンテキスト デバイス

コンテキスト デバイスとは現行の HTTP 要求に関連付けられたデバイスです。`HttpRequestBegin` パイプラインの中の、デバイスの解決プロセッサ

(`Sitecore.Pipelines.HttpRequest.DeviceResolver`) がコンテキスト デバイスを設定します。コンテキスト アイテムのコンテキスト 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 コンポーネントを使って、レイアウト エンジンが要求を処理します。

`Sitecore.Data.Items.DeviceItem` クラスと `Sitecore.Context.Device` プロパティを使ってコンテキスト デバイスにアクセスすることができます。たとえば:

```
Sitecore.Data.Items.DeviceItem contextDevice = Sitecore.Context.Device;
```

コンテキスト デバイス プロパティが定義されていない場合、デバイスの解決プロセッサは有効なデバイスを指定する次のリストから最初の値をコンテキスト デバイスに設定します:

- `sc_device` クエリ文字列パラメーター
- コンテキスト サイトの `device` 属性
- URL クエリ文字列パラメーターまたは現行の HTTP 要求に関連付けられた Web クライアント ユーザーエージェントに合致するコンテキスト データベースの最初のデバイス
- コンテキスト サイトの `defaultDevice` 属性
- [データ] セクションのデフォルトのチェックボックス フィールドが選択されている最初のデバイス

メモ

管理 Web サイト定義の `device` 属性を使って、`sc_device` クエリ文字列パラメーターを含まない Web サイトに関連付けられたすべての要求に対して、そのデバイスがコンテキスト デバイスとなるようにすることができます。`defaultDevice` 属性を使って別の管理 Web サイトには別のデフォルト デバイスを指定します。

2.6.1 サンプル コード: コンテキスト デバイスの設定

次のサンプル コードを元にして要求処理パイプラインを実装することができます。このサンプル コードでは、Web クライアント ユーザー エージェント文字列がいくつかの一般的な RSS リーダーの 1 つのユーザー エージェントと合致する場合に、コンテキスト デバイスを `Feed` という名前のデバイスに設定しています。

```
namespace Sitecore.Sharedsource.Pipelines.HttpRequest
{
    public class FeedDeviceResolver
    {
        public void Process(Sitecore.Pipelines.HttpRequest.HttpRequestArgs args)
        {
            System.Web.HttpContext httpContext = System.Web.HttpContext.Current;

            if (httpContext == null || httpContext.Request == null
                || httpContext.Request.UserAgent == null || Sitecore.Context.Database == null)
            {

```

```
        return;  
    }  
  
    string agent = httpContext.Request.UserAgent;  
  
    if (agent.Contains("feedfetcher")  
        || agent.Contains("newsgatoronline")  
        || agent.Contains("bloglines")  
        || agent.Contains("yahoofeedseeker"))  
    {  
        Sitecore.Data.Items.DeviceItem device =  
            Sitecore.Context.Database.Resources.Devices["Feed"];  
  
        if (device != null)  
        {  
            Sitecore.Context.Device = device;  
        }  
    }  
}
```

web.config の /configuration/sitecore/pipelines/httpRequestBegin 要素の中でデバイス解決の前にプロセッサーを挿入します。

```
<httpRequestBegin>  
...  
  <processor  
    type="Sitecore.Sharedsource.Pipelines.HttpRequest.FeedDeviceResolver, Assembly" />  
  <processor type="Sitecore.Pipelines.HttpRequest.DeviceResolver, Sitecore.Kernel" />  
...
```

Sitecore RSS 機能についての詳細は、『[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。

RSS API に関する詳細は、『[コンテンツ API クックブック](#)』を参照してください。

2.7 コンテキスト ドメイン

コンテキスト ドメインとはコンテキスト サイトに関連付けられた Sitecore のセキュリティ ドメインです。Sitecore は要求処理パイプラインを使用せずにコンテキスト ドメインを動的に決定します。各管理 Web サイトはそのサイトのデフォルトのコンテキスト ドメインを指定することができます。コンテキスト サイトはデフォルトのコンテキスト ドメインを定義します。Sitecore のユーザー インターフェイスと URL パラメーターはコンテキスト サイトに関連付けられたドメインと異なるコンテキスト ドメインを指定することができます。Sitecore のユーザー インターフェイスと URL のクエリ文字列パラメーターはコンテキスト サイトに指定されたデフォルトのコンテキスト ドメインと異なるコンテキスト ドメインを指定することができます。

セキュリティ API に関する詳細は、『[セキュリティ API クックブック](#)』を参照してください。

Sitecore.Security.Domains.Domain クラスと Sitecore.Context.Domain プロパティを使ってコンテキスト ドメインにアクセスすることができます。たとえば：

```
Sitecore.Security.Domains.Domain contextDomain = Sitecore.Context.Domain;
```

レジストレーションやログインのフォームなどのセキュリティ API を使用するプレゼンテーション コンポーネントは多くの場合コンテキスト ドメインにアクセスします。

2.8 コンテキスト ユーザー

コンテキスト ユーザーとは現行の HTTP 要求に関連付けられたセキュリティ ユーザーです。

`HttpRequest.Begin` パイプラインの中の、ユーザーの解決プロセッサ

(`Sitecore.Pipelines.HttpRequest.UserResolver`) がコンテキスト ユーザーを設定します。ユーザーが認証されていない場合は、コンテキスト ユーザーはコンテキスト ドメインの匿名ユーザーです。

`Sitecore.Security.Accounts.User` クラスと `Sitecore.Context.User` プロパティを使ってコンテキスト ユーザーにアクセスすることができます。たとえば：

```
Sitecore.Security.Accounts.User contextUser = Sitecore.Context.User;
```

`Sitecore.Context.IsLoggedIn()` メソッドを使ってユーザーが認証されているかどうかを知ることができます。たとえば：

```
if (Sitecore.Context.IsLoggedIn)
{
    //TODO: handle case that user is authenticated
}
else
{
    //TODO: handle case that user is not authenticated
}
```

2.9 コンテキストの生の URL

コンテキストの生の URL は現行の HTTP 要求のパスとクエリ文字列パラメーターへのアクセス（プロトコルとホストネームを除く）を提供します。Sitecore は `HttpRequest.Begin` 要求処理パイプラインを使わずに生の URL を決定します。たとえば、URL `http://localhost/path/item.aspx?q=v` では、コンテキストの生の URL は `/path/item.aspx?q=v` です。

`Sitecore.Context.RawUrl` プロパティを使ってコンテキストの生の URL にアクセスすることができます。たとえば：

```
string rawUrl = Sitecore.Context.RawUrl;
```

いろいろな条件をロギングする場合に生の URL を含めることができます。たとえば：

```
string message = String.Format("{0} : {1}", GetType(), Sitecore.Context.RawUrl);
Sitecore.Diagnostics.Log.Info(message, this);
```

`Sitecore.Web.WebUtil.GetLocalPath()` メソッドを `Sitecore.Context.RawUrl` プロパティと一緒に使用して生の URL のパスにアクセスすることができます。たとえば：

```
string rawUrl = Sitecore.Context.RawUrl;
string path = Sitecore.Web.WebUtil.GetLocalPath(rawUrl);
```

`Sitecore.Web.WebUtil.ParseQueryString()` メソッドを `Sitecore.Context.RawUrl` プロパティと一緒に使用してクエリ文字列をコレクションに解析することができます。たとえば、`key` という名前の URL クエリ文字列パラメーターの値を読み出す方法：

```
string rawUrl = Sitecore.Context.RawUrl;
Sitecore.Collections.SafeDictionary<string> queryString =
    Sitecore.Web.WebUtil.ParseQueryString(rawUrl);
string value = queryString["key"];

if (value == null)
{
    output.Write("//TODO: handle case that query string does not include parameter");
}
else if (String.IsNullOrEmpty(value))
{
    output.Write("//TODO: handle case that parameter is empty");
}
else
{
    output.Write("//TODO: process value");
}
```


2.10 コンテキスト ページ モード

ページ モードは現行の HTTP 要求に有効な Sitecore の機能を示します。Sitecore は要求処理パイプラインを使用せずにページ モードを動的に決定します。プレゼンテーション コンポーネントは異なるページ モードで異なる出力を生成することができます。たとえば、パブリッシュされたサイトにおける場合、ページ エディターを使ってナビゲート中の場合、またはインライン編集の場合、またはデバッガーにおける場合、などです。

ページ モードについての詳細は、『[クライアント構成クックブック](#)』を参照してください。

`Sitecore.Context.PageMode` プロパティを使ってページ モードを知ることができます。たとえば、Web クライアントがページ エディターにいるかどうかを知る方法:

```
if (Sitecore.Context.PageMode.IsPageEditor)
{
    //TODO: handle case that user is in the Page Editor
}
```

2.10.1 サンプル コード: メタデータ フィールドの公開

メタデータ フィールドは場合によって、パブリッシュされた Web サイトで表示されるページ要素を持たないことがあります。「FieldRenderer Web コントロール」のセクションで記載されているように FieldRenderer Web コントロールを使って、ユーザーがページ エディターで編集している場合のみメタデータ フィールドのためのインライン編集コントロールを有効にすることもできます。たとえば、コンテキスト アイテムの [タイトル] フィールドを編集するには、次のような FieldRenderer Web コントロールをレイアウトまたはサブレイアウトファイルに追加します:

```
<sc:FieldRenderer runat="server" id="fldTitle" FieldName="title" visible="false" />
```

レイアウトまたはサブレイアウトのコード ファイルで `Page_Load()` メソッドを使って、ユーザーがページ エディターで編集している場合に FieldRenderer Web コントロールを有効にすることができます。:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Sitecore.Context.PageMode.IsPageEditorEditing)
    {
        fldTitle.Visible = true;
    }
}
```

Chapter 3

アイテムにアクセスする

この章ではアイテムにアクセスするために使用する API についての情報を解説します。まずアイテムへのアクセスの概要を説明し、続いてアイテムからのデータを使って ASP.NET コントロールを生成すること、アイテムの子孫と祖先にアクセスすることを説明し、最後にアイテムの並べ替えについて説明します。

この章には次のセクションがあります。

- アイテムへのアクセスの概要
- アイテムの値を使って ASP.NET コントロールを生成する
- アイテムの子孫にアクセスする方法
- アイテムの祖先にアクセスする方法
- アイテムを並べ替える

3.1 アイテムへのアクセスの概要

このセクションでは、アイテムにアクセスするために使用する API の概要を説明します。

アイテムにアクセスするための API についての詳細は、『[コンテンツ API クックブック](#)』を参照してください。

3.1.1 コンテンツ アイテムのわかりやすい URL にアクセスする方法

`Sitecore.Links.LinkManager.GetItemUrl()` メソッドを使って、コンテンツ アイテムのわかりやすい URL にアクセスすることができます。コンテキスト アイテムのわかりやすい URL にアクセスする方法:

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
string url = Sitecore.Links.LinkManager.GetItemUrl(contextItem);
```

動的なリンク管理に関する詳細は、『[コンテンツ API クックブック](#)』を参照してください。

3.1.2 メディア アイテムのわかりやすい URL にアクセスする方法

`Sitecore.Resources.Media.MediaManager.GetMediaUrl()` メソッドを使って、メディア アイテムのわかりやすい URL にアクセスすることができます。たとえば、コンテキスト データベースの `ImageField` という名前のイメージ フィールドで参照されているメディア アイテムのわかりやすい URL にアクセスする方法:

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
Sitecore.Data.Fields.ImageField image = contextItem.Fields["imagefield"];

if (image.MediaItem != null)
{
    string url = Sitecore.StringUtil.EnsurePrefix('/',
        Sitecore.Resources.Media.MediaManager.GetMediaUrl(image.MediaItem));
}
```

メディア URL の管理に関する詳細は、『[コンテンツ API クックブック](#)』を参照してください。

3.1.3 アイテムにデバイスのためのレイアウト詳細があるかどうかを知る方法

`Sitecore.Data.Items.Item.Visualization.GetLayout()` メソッドを使って、あるアイテムに、あるデバイスのためのレイアウト詳細があるかどうかを知ることができます。

`Sitecore.Data.Items.Item.Visualization.GetLayout()` メソッドは、アイテムが指定されたデバイスのためのレイアウト詳細を持っていない場合には `Null` を返します。たとえば、コンテキスト デバイスのためのレイアウト詳細を含むコンテキスト アイテムの子のそれぞれへのリンクのリストを生成するには:

```
System.Text.StringBuilder markupBuilder = new System.Text.StringBuilder();
foreach (Sitecore.Data.Items.Item child in Sitecore.Context.Item.Children)
{
    Sitecore.Data.Items.LayoutItem layout =
        child.Visualization.GetLayout(Sitecore.Context.Device);
    if (layout != null)
    {
        string url = Sitecore.Links.LinkManager.GetItemUrl(child);
        markupBuilder.AppendFormat("<li><a href=\"\"{0}\"\">{1}</a></li>", url, child.Name);
    }
}
```

```
if (markupBuilder.Length > 0)
{
    markupBuilder.Insert(0, "<ul>");
    markupBuilder.Append("</ul>");
    string markup = markupBuilder.ToString();
}
```

3.2 アイテムの値を使って ASP.NET コントロールを生成する

このセクションで説明する API を使うと、アイテムのデータを使ってドロップダウン リストなどの ASP.NET コントロールを生成することができます。

3.2.1 アイテムの子を ASP.NET コントロールにバインドする方法

`Sitecore.Data.Items.Item.Children` プロパティを使って、アイテムの子を ASP.NET コントロールにバインドすることができます。たとえば、アイテムの子の名前を選択肢として使う ASP.NET のドロップダウン リストを生成するには、これらのアイテムの ID をそれらの選択肢の値として使用して、次のようなドロップダウン リスト コントロールをレイアウトまたはサブレイアウトのファイルに追加します：

```
<asp:dropdownlist runat="server" id="lstDropDown" />
```

レイアウトまたはサブレイアウト ファイルの `Page_Load()` メソッドのドロップダウン リストのためのデータ バインドを実装します：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
        lstDropDown.DataSource = contextItem.Children;
        lstDropDown.DataTextField = "name";
        lstDropDown.DataValueField = "id";
        lstDropDown.DataBind();
    }
}
```

アイテムのリストをコントロールにバインドする方法

`Sitecore.Data.Items.Item` オブジェクトのリストを ASP.NET コントロールにバインドすることができます。たとえば、`Common/Folder` データ テンプレートに基づくコンテキスト データベースの `/Sitecore/Content` アイテムのすべての子孫の名前を選択肢として使う ASP.NET のドロップダウン リストを生成するには、これらのアイテムの ID をそれらの選択肢の値として使用して、次のようなドロップダウン リスト コントロールをレイアウトまたはサブレイアウトのファイルに追加します：

```
<asp:dropdownlist runat="server" id="lstDropDown" />
```

レイアウトまたはサブレイアウト ファイルの `Page_Load()` メソッドのドロップダウン リストのためのデータ バインドを実装します：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        string query = String.Format(
            "/sitecore/content/*[@@templateid='{0}']", Sitecore.TemplateIDs.Folder);
        Sitecore.Data.Items.Item[] queried = Sitecore.Context.Database.SelectItems(query);

        if (queried != null)
        {
            lstDropDown.DataSource = queried;
            lstDropDown.DataTextField = "name";
            lstDropDown.DataValueField = "id";
            lstDropDown.DataBind();
        }
    }
}
```

```
    }
  }
}
```

メモ

この例では子孫軸 ("///") を使っていますが、これは子孫の数の増加に従って非効率になります。多くのアイテムを起動するソリューションでは、子孫軸を使用しないような代替の情報体系を検討してください。

3.2.2 アイテムをコントロールにバインドする際のフィールド値の使用法

アイテムのリストを ASP.NET コントロールにバインドする際にフィールド値を使うことができます。たとえば、コンテキスト アイテムの子の [タイトル] フィールドの値を選択肢として使う ASP.NET のドロップダウンリストを生成するには、これらのアイテムの ID をそれらの選択肢の値として使用して、次のようなドロップダウン リスト コントロールをレイアウトまたはサブレイアウトのファイルに追加します：

```
<asp:dropdownlist runat="server" id="lstDropDown" />
```

レイアウトまたはサブレイアウト ファイルの `Page_Load()` メソッドのドロップダウン リストを生成することができます：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;

        foreach (Sitecore.Data.Items.Item child in contextItem.Children)
        {
            System.Web.UI.WebControls.ListItem option =
                new System.Web.UI.WebControls.ListItem(child.Name, child.ID.ToString());
            lstDropDown.Items.Add(option);
        }
    }
}
```

3.2.3 アイテムのデータを使って ASP.NET リピーターを生成する

`Sitecore.Data.Items.Item` オブジェクトのリストからのデータを使って ASP.NET リピーターを生成することができます。たとえば、`Sitecore.Data.Items.Item.Name` プロパティとコンテキスト アイテムの子のそれぞれの [タイトル] フィールドの値をリストする HTML テーブルを生成するには、次のようなリピーターをレイアウトまたはサブレイアウトファイルに追加します：

```
<table border="1">
  <tr>
    <th>Item</th>
    <th>Title</th>
  </tr>
  <asp:Repeater ID="repeater" runat="server">
    <ItemTemplate>
      <tr>
        <td>
          <asp:Label runat="server" ID="Name"
            Text='<#DataBinder.Eval(Container.DataItem, "Name") %>'></asp:Label>
        </td>
        <td>
```

```
<asp:Label runat="server" ID="Title"
Text='<%#DataBinder.Eval(Container.DataItem,@"["title"]")%>'></asp:Label>
</td>
</tr>
</ItemTemplate>
</asp:Repeater>
</table>
```

レイアウトまたはサブレイアウト ファイルの `Page_Load()` メソッドのリピーターを生成することができます :

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        repeater.DataSource = Sitecore.Context.Item.Children;
        repeater.DataBind();
    }
}
```


3.3 アイテムの子孫にアクセスする方法

このセクションで説明するように、再帰的なメソッドを使ってアイテムの子孫にアクセスすることができます。

3.3.1 サンプル コード: サイト マップ Web コントロール

次のサンプル コードのように Web コントロールを実装することができます。このサンプル コードではコンテキスト サイトのホーム アイテムの子孫にアクセスし、簡単なデータ駆動型のサイト マップを生成します。

```
using System;

namespace Sitecore.Sharedsource.Web.UI.WebControls
{
    public class SiteMap : Sitecore.Web.UI.WebControl
    {
        protected override void DoRender(System.Web.UI.HtmlTextWriter output)
        {
            Sitecore.Data.Items.Item home =
                Sitecore.Context.Database.GetItem(Sitecore.Context.Site.StartPath);
            this.SiteMapStep(home, output, 1);
        }

        private void SiteMapStep(
            Sitecore.Data.Items.Item item, System.Web.UI.HtmlTextWriter output, int level)
        {
            if (this.HasVisibleChildren(item))
            {
                output.Write("<ul>");

                foreach (Sitecore.Data.Items.Item child in item.Children)
                {
                    if (IsVisible(child))
                    {
                        string markup = String.Format(
                            @"<li class="{0}"><a href="{1}">{2}</a>",
                            "sitemap_level " + level,
                            Sitecore.Links.LinkManager.GetItemUrl(child),
                            child.Name);
                        output.Write(markup);
                        this.SiteMapStep(child, output, level + 1);
                        output.Write("</li>");
                    }
                }

                output.Write("</ul>");
            }
        }

        private bool IsVisible(Sitecore.Data.Items.Item item)
        {
            return item.Versions.Count > 0
                && item.Visualization.GetLayout(Sitecore.Context.Device) != null;
        }

        private bool HasVisibleChildren(Sitecore.Data.Items.Item item)
        {
            foreach (Sitecore.Data.Items.Item child in item.Children)
            {
                if (this.IsVisible(child))
                {
                    return true;
                }
            }

            return false;
        }
    }
}
```

```
}
```

このサイト マップ Web コントロールはコンテキスト サイトのホーム アイテムを再帰メソッド

`SiteMapStep()` に渡します。そのアイテムにレイアウト詳細とコンテキスト言語のバージョンのデータをもつ子がある場合、`SiteMapStep()` メソッドはそれらの子のそれぞれにリンクし、その子が同じ基準によって参照できる子をもつ場合、その子を再帰的に `SiteMapStep()` に渡します。

Web コントロールの実装についての詳細は、『[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

この例はサイトを閲覧する人のためのビジュアルなサイトマップを作成します。検索エンジンのために Web サイトの構造を記述するために使われる検索エンジン サイト マップではありません。

3.4 アイテムの祖先にアクセスする方法

このセクションで説明するように、`Sitecore.Data.Items.Item.Axes.GetAncestors()` メソッドと `Sitecore.Data.Items.Axes.IsDescendantOf()` メソッドを使ってアイテムの祖先を使用することができます。

メモ

`Sitecore.Data.Items.Item.Parent` プロパティを使ってアイテムの祖先に再帰的にアクセスすることもできます。

3.4.1 サンプル コード: ブレッドクラム Web コントロール

ブレッドクラムは Web サイトの情報体系の中で現行のページの場所を示します。多くの場合は情報体系の中でそのアイテムより上位のレベルのそれぞれへのリンクを持っています。次のサンプル コードのように Web コントロールを実装することができます。このサンプル コードではアイテムの祖先にアクセスして簡単なブレッドクラムを生成します。

```
using System;

namespace Sitecore.Sharedsource.Web.UI.WebControls
{
    public class Breadcrumb : Sitecore.Web.UI.WebControl
    {
        protected override void DoRender(System.Web.UI.HtmlTextWriter output)
        {
            Sitecore.Data.Items.Item home =
                Sitecore.Context.Database.GetItem(Sitecore.Context.Site.StartPath);

            foreach (Sitecore.Data.Items.Item item in
                Sitecore.Context.Item.Axes.GetAncestors())
            {
                if (item.ID == home.ID || item.Axes.IsDescendantOf(home))
                {
                    if (item.Visualization.GetLayout(Sitecore.Context.Device) != null
                        && item.Versions.Count > 0)
                    {
                        output.Write(String.Format("<a href=\"\"{0}\"\">{1}</a>",
                            Sitecore.Links.LinkManager.GetItemUrl(item), item.Name));
                    }
                    else
                    {
                        output.Write(item.Name);
                    }
                }

                output.Write(" &gt; ");
            }

            output.Write(Sitecore.Context.Item.Name);
        }
    }
}
```

ブレッドクラム Web コントロールはコンテキスト サイトのホーム アイテムから始まります。ブレッドクラムはコンテキスト アイテムの祖先のそれぞれを含みます。それらはホーム アイテムかまたはホーム アイテムの子孫であり、階層構造の上から下の順に並んでいます。アイテムにレイアウト詳細とコンテキスト言語のバージョンのデータがある場合、ブレッドクラムはそのアイテムにリンクします。ない場合にはブレッドクラムはアイテムの名前を

表示します。ブレッドキラムは各ステップの間にスペース要素を含みます。最後はコンテキスト アイテムの名前です。

Web コントロールの実装についての詳細は、『[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。

3.4.2 サンプル コード: 整形されたナビゲーション Web コントロール

次のサンプル コードのように Web コントロールを実装することができます。このサンプル コードではナビゲーション リンクがコンテキスト アイテムまたはその祖先の 1 つにリンクしている場合、ナビゲーション リンクを他と異なるように整形します。

```
using System;

namespace Sitecore.Sharedsource.Web.UI.WebControls
{
    public class TopNav : Sitecore.Web.UI.WebControl
    {
        protected override void DoRender(System.Web.UI.HtmlTextWriter output)
        {
            Sitecore.Data.Items.Item home =
                Sitecore.Context.Database.GetItem(Sitecore.Context.Site.StartPath);
            output.Write("<ul>");

            foreach (Sitecore.Data.Items.Item child in home.Children)
            {
                if (child.Visualization.GetLayout(Sitecore.Context.Device) != null
                    && child.Versions.Count > 0)
                {
                    string cssClass = "topnav default";

                    if (Sitecore.Context.Item.ID == child.ID)
                    {
                        cssClass = "topnav_selected";
                    }
                    else if (Sitecore.Context.Item.Axes.IsDescendantOf(child))
                    {
                        cssClass = "topnav current";
                    }

                    string markup = String.Format(
                        @"<li class="{0}"><a href="{1}">{2}</a></li>",
                        cssClass,
                        Sitecore.Links.LinkManager.GetItemUrl(child),
                        child.Name);
                    output.Write(markup);
                }
            }

            output.Write("</ul>");
        }
    }
}
```

このナビゲーション Web コントロールは、レイアウト詳細とコンテキスト言語のバージョンデータを持つコンテキスト サイトのホームアイテムの子のそれぞれへのリンクのリストを生成します。ユーザーがアイテムまたはそのアイテムの子孫を要求した場合には、このコントロールは別の CSS クラスを適用します。

Web コントロールの実装についての詳細は、『[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。

3.5 アイテムを並べ替える

Sitecore は各アイテムの子を、そのアイテムに関連付けられた "子の並べ替えのルール" に従って並べ替えます。各アイテムには別の "子の並べ替えのルール" を選択することができます。Sitecore はいくつかの "子の並べ替えのルール" を提供しています。アイテムに "子の並べ替えのルール" を指定しなかった場合、Sitecore はデフォルトの "子の並べ替えのルール" を適用し、ユーザーはアイテムを手動で並べ替えることができます。

メモ

"子の並べ替えのルール" をアイテムまたはデータ テンプレートの標準値に適用した場合、手動でアイテムを並べ替えても "子の並べ替えのルール" を上書きすることはできません。

メモ

アイテムの比較機能はアイテムの名前の先頭の数字を文字列または数値として解釈することができます。これによって、10 から始まる名前のアイテムを 2 から始まる名前のアイテムの前に並べ替えることも後に並べ替えることも可能です。

メモ

各種のシステム アイテムの名前は 2 つのアンダースコア文字 ("__") で始まります。アイテムの比較機能は並べ替えによって 2 つのアンダースコア文字で始まる名前をリストの最初または最後にすることができます。

3.5.1 子の並べ替えのルール

Sitecore は次の "子の並べ替えのルール" を提供しています。

ルール	ロジック
Created	各アイテムの最新のバージョンの作成日時によりアイテムを並べ替える。並べ替えによって最も古いバージョンのアイテムが最後になる。
Default	名前によりアルファベット順にアイテムを並べ替える。先頭の数字は数値と解釈しない。並べ替えによって先頭のアンダースコアは最後になる。これはデフォルトの "子の並べ替えのルール"。
Display Name	表示名によりアルファベット順にアイテムを並べ替える。先頭の数字は数値と解釈する。並べ替えによって先頭のアンダースコアは最後になる。

ルール	ロジック
Logical	名前によりアルファベット順にアイテムを並べ替える。先頭の数字は数値と解釈する。並べ替えによって先頭のアンダースコアは最初になる。
Reverse	名前により逆アルファベット順にアイテムを並べ替える。先頭の数字は数値と解釈しない。並べ替えによって先頭のアンダースコアは最初になる。
Updated	最後に更新された日時でアイテムを並べ替える。並べ替えによって最近更新されたアイテムが最初になる。
[標準にリセット]	アイテムの "子の並べ替えのルール" をデータ テンプレートの標準値で定義されたものにリセットするか、またはデフォルトの "子の並べ替えのルール" にリセットする。

3.5.2 子の並べ替えのルールの選択方法

アイテムの子の並べ替えを制御するために "子の並べ替えのルール" を選択する方法:

1. **テンプレート マネージャー**または**コンテンツ エディター**で標準値アイテムまたは個々のアイテムを選択します。
2. [ホーム] タブの、[並べ替え] グループで、[子の並べ替えのルールのダイアログの起動] ([並べ替え] グループの一番下の矢印を含む四角のアイコン) をクリックします。[子の並べ替えのルール] のダイアログが表示されます。
3. [子の並べ替えのルール] のダイアログで、子の並べ替えのルールを選択します。

3.5.3 カスタムのアイテムの比較機能の実装方法

カスタムのアイテムの比較機能を実装し、`Sitecore.Data.Items.Item` オブジェクトのリストを並べ替えることができます。比較機能を "子の並べ替えのルール" として構成し、自分のコードから比較機能呼び出すことができます。カスタムのアイテムの比較機能を作成するには、

`System.Collections.Generic.IComparer`

<`Sitecore.Data.Items.Item`> インターフェースの `Compare()` メソッド を実装します。

サンプル コード: アイテムをフィールド値で並べ替える

アイテムを共通のフィールド値によって並べ替えるカスタムのアイテムの比較機能を実装することができます。

3.5.4 カスタムの子の並べ替えのルールの実装方法

カスタムの子の並べ替えのルールを実装する方法:

1. 「カスタムのアイテムの比較機能の実装方法」のセクションに記載されているように、カスタムのアイテムの比較機能を実装します。

たとえば、Title という名前のフィールドでアイテムを並べ替えるには：

```
namespace Sitecore.Sharedsource.Data.Comparers
{
    public class TitleComparer : Sitecore.Sharedsource.Data.Comparers.ItemFieldComparer
    {
        public TitleComparer() : base("title")
        {
        }
    }
}
```

2. **コンテンツ エディター**で /Sitecore/System/Settings/Subitems Sorting アイテムを選択します。
3. /System/Child Sorting データテンプレートを使って子の並べ替えのルール定義アイテムを挿入します。アイテムの名前には、カスタムのアイテムの比較機能のクラスまたはカスタムのアイテムの比較機能が並べ替えに使うプロパティの名前を使用します。
4. **コンテンツ エディター**で子の並べ替えのルール定義アイテムを選択します。
5. 新規の子の並べ替えのルール定義アイテムで、[データ] セクションの [種類] フィールドで、カスタムのアイテムの比較機能のクラスのシグネチャーを入力します：

```
Sitecore.Sharedsource.Data.Comparers.TitleComparer, Assembly
```

アイテムに子の並べ替えのルールの選択についての追加情報は、「子の並べ替えのルールの選択方法」のセクションを参照してください。

3.5.5 アイテムのリストを並べ替える方法

アイテムの比較機能と `Array.Sort()` メソッドを使って、アイテムのリストを並べ替えることができます。次のサンプル コードを元にしてソリューションを実装することができます。このサンプル コードでは「カスタムの子の並べ替えのルールの実装方法」のセクションで記載した Title フィールド比較機能を使って、コンテキスト アイテムの子を並べ替えます。

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
Sitecore.Data.Items.Item[] items = contextItem.Children.ToArray();
Sitecore.Sharedsource.Data.Comparers.TitleComparer titleComparer = new
    Sitecore.Sharedsource.Data.Comparers.TitleComparer();
Array.Sort(items, titleComparer);
```

「サンプル コード: 最新のアイテム Web コントロール」のセクションで記載したカスタムのアイテムの比較機能を起動した後で、`Array.Reverse()` メソッドを起動して並べ替え順を逆にすることができます。

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
Sitecore.Data.Items.Item[] items = contextItem.Children.ToArray();
Sitecore.Sharedsource.Data.Comparers.TitleComparer titleComparer = new
    Sitecore.Sharedsource.Data.Comparers.TitleComparer();
Array.Sort(items, titleComparer);
Array.Reverse(items);
```

または、既存の比較機能を元にしてカスタムのアイテムの比較機能を実装することができます。たとえば、Title フィールドの値によって逆アルファベット順にアイテムを並べ替える比較機能を実装するには：

```
namespace Sitecore.Sharedsource.Data.Comparers
{
    public class ReverseTitleComparer :
        Sitecore.Sharedsource.Data.Comparers.TitleComparer
    {
        public override int Compare(Sitecore.Data.Items.Item xItem,
            Sitecore.Data.Items.Item yItem)
        {
            return base.Compare(xItem, yItem) * -1;
        }
    }
}
```

ヒント

「子の並べ替えのルール」のセクションで記載された子の並べ替えのルールで使われているアイテムの比較機能の 1 つを使うためには、/Sitecore/System/Settings/Subitems Sorting アイテムの下に対応する子の並べ替えのルールの定義アイテムの、[データ] セクションの [種類] フィールドで指定されたクラスを使います。

3.5.6 サンプル コード: 最新のアイテム Web コントロール

次のサンプル コードを元にして Web コントロールを実装することができます。このサンプル コードでは、レンダリングのデータ ソース アイテムの子孫である特定のデータ テンプレートに基づいた最も新しいアイテムにアクセスします。

```
using System;

namespace Sitecore.Sharedsource.Web.UI.WebControls
{
    public class LatestArticles : Sitecore.Web.UI.WebControl
    {
        private string _article_Template = null;
        private string _sort_Field = null;
        private int _maxArticles = -1;

        public string ArticleTemplate
        {
            get
            {
                return this._article_Template;
            }
            set
            {
                this._article_Template = value;
            }
        }

        public string SortField
        {
            get
            {
                return this._sort_Field;
            }
            set
            {
            }
        }
    }
}
```



```
        this._sort_Field = value;
    }
}

public int MaxArticles
{
    get
    {
        return this.maxArticles;
    }

    set
    {
        this.maxArticles = value;
    }
}

protected override void DoRender(System.Web.UI.HtmlTextWriter output)
{
    Sitecore.Diagnostics.Assert.IsNotNull(
        this.ArticleTemplate,
        "Specify the ArticleTemplate property");
    Sitecore.Diagnostics.Assert.IsNotNullOrEmpty(
        this.SortField,
        "Specify the SortField property");
    Sitecore.Diagnostics.Assert.IsTrue(
        this.MaxArticles > 0,
        "Specify the MaxArticles property");
    Sitecore.Data.Items.Item dataSource = GetItem();
    Sitecore.Diagnostics.Assert.IsNotNull(
        dataSource,
        "Invalid data source item");
    Sitecore.Data.Items.TemplateItem articleTemplate =
        Sitecore.Context.Database.Templates[this.ArticleTemplate];
    Sitecore.Diagnostics.Assert.IsNotNull(
        articleTemplate,
        "Invalid ArticleTemplate " + this.ArticleTemplate);
    string query = String.Format(
        "{0}/*[@@templateid='{1}']",
        dataSource.Paths.FullPath,
        articleTemplate.ID);
    Sitecore.Data.Items.Item[] queried =
        Sitecore.Context.Database.SelectItems(query);

    if (!this.WriteItems(queried, output))
    {
        string message = String.Format(
            "{0} : No items under {1} based on {2}",
            GetType(),
            dataSource.Paths.FullPath,
            articleTemplate.FullName);
        Sitecore.Diagnostics.Log.Warn(message, this);
    }
}

private bool WriteItems(
    Sitecore.Data.Items.Item[] items,
    System.Web.UI.HtmlTextWriter output)
{
    if (items != null && items.Length > 0 && MaxArticles > 0)
    {
        Sitecore.Sharedsource.Data.Comparers.ItemFieldComparer comparer =
            new Sitecore.Sharedsource.Data.Comparers.ItemFieldComparer(SortField);
        Array.Sort(items, comparer);
        Array.Reverse(items);

        if (items.Length > this.MaxArticles)
        {
            Array.Resize(ref items, this.MaxArticles);
        }

        output.Write("<ul>");
    }
}
```

```
foreach (Sitecore.Data.Items.Item article in items)
{
    string markup = String.Format(
        @"<li><a href="{0}">{1}</a> ({2})</li>",
        Sitecore.Links.LinkManager.GetItemUrl(article),
        article.Name,
        article.Statistics.Created.ToString("f"));
    output.Write(markup);
}

output.Write("</ul>");
return true;
}

return false;
}
```

Web コントロール LatestArticles がその挙動を制御するいくつかのプロパティを公開します。

Web コントロール LatestArticle の SortField プロパティは並べ替えに使うフィールドの名前または ID を含みます。たとえば、__Created を指定して、バージョンの作成日時で並べ替えたり、__Updated を指定して言語の最終更新日時で並べ替えたりすることができます。

Web コントロールを実装してデータ ソースを含むパラメーターをコントロールに渡す方法については、『[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。

ヒント

並べ替えのために [日付] フィールドを作成したら、そのフィールドの標準値を \$now に設定し、そのフィールドの初期値をそのアイテムの作成日時に設定することができます。

Date または Datetime フィールドの標準値への \$now などの値の入力と、生のフィールド値の参照については、『[クライアント構成クックブック](#)』を参照してください。

Web コントロール LatestArticle の ArticleTemplate プロパティは Article データ テンプレートのフルネームを含みます。たとえば、Sample/Sample Item データ テンプレートを元にアイテムをリストするには、ArticleTemplate を Sample/Sample Item に設定します。

Web コントロール LatestArticle の MaxArticles プロパティはリストする記事 (article) の最大数を含んでいます。たとえば、最新の 5 つの記事にアクセスするには、MaxArticles を 5 にセットします。

Web コントロール LatestArticle は Sitecore クエリを使って ArticleTemplate プロパティに指定されたデータ テンプレートに基づくデータ ソース アイテムのすべての子孫を読み出します。(レンダリングのデフォルトのデータ ソース アイテムはコンテキスト アイテムです) 次にコントロールは、「サンプル コード: アイテムをフィールド値で並べ替える」のセクションに記載されたアイテムの比較機能を使って、記事を SortField プロパティに指定されたフィールドによって並べ替え、リストを逆順にし、MaxArticles で指定された制限を超える場合はリストのサイズを調整し、各記事へのリンクをレンダリングします。

レンダリングにデータ ソースを指定する方法については、『[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

この例では子孫軸 (“//”) を使っていますが、これは子孫の数の増加に従って非効率になります。多くのアイテムを起動するソリューションでは、子孫軸を使用しないような代替の情報体系を検討してください。たとえば、年月を表すフォルダーにニュース記事を保存し、子孫軸を使用する代わりに、最新のフォルダーのみにアクセスするコードを書きます。

Chapter 4

フィールドにアクセスする

この章ではフィールドにアクセスするために使用する API についての情報を解説します。まずフィールドへのアクセスの概要を説明し、続いて `renderField` パイプライン、`FieldRenderer` Web コントロール、フィールド タイプ Web コントロールを説明し、最後にフィールドへのアクセスのサンプルコードを説明します。

この章には次のセクションがあります。

- フィールドへのアクセスの概要
- `renderField` パイプライン
- `FieldRenderer` Web コントロール
- フィールド タイプ Web コントロール
- フィールドへのアクセスのサンプル

4.1 フィールドへのアクセスの概要

`Sitecore.Data.Items.Item.Fields` コレクション プロパティを使ってフィールド値を文字列としてアクセスすることができます。または別のタイプのフィールドには `Sitecore.Data.Fields` 名前空間の特定のクラスを使うことができます。

リッチ テキスト エディター (RTE) フィールドの RAW 値は 1 つのアイテムから別のアイテムへの動的なリンクを表す GUID (Global Unique Identifiers) を含む場合があります。プレゼンテーション コンポーネントは、生の RTE フィールド値の動的なリンクをわかりやすい URL に変換するコンストラクトを使用する必要があります。

動的なリンクとわかりやすい URL に関する詳細は、『[コンテンツ API クックブック](#)』を参照してください。

インライン編集をサポートするフィールドタイプの値をレンダリングするプレゼンテーション コンポーネントは、インライン編集コントロールを生成する API を使用する必要があります。このセクションに記載されている API を使って、動的な URL を変換したり、インライン編集コントロールをレンダリングすることができます。

メモ

すべてのフィールドのすべての使用にインライン編集コントロールを有効にする必要はありません。インライン編集を有効にしたい場合のみ、インライン編集機能を使用します。

フィールドにアクセスするための API についての詳細は、『[コンテンツ API クックブック](#)』を参照してください。

4.1.1 RTE フィールドの動的なリンクをわかりやすい URL に変換する方法

`Sitecore.Links.LinkManager.GetItemUrl()` メソッドを使ってインライン編集を有効化することなく、RTE フィールドの動的なリンクをわかりやすい URL に変換することができます。たとえば、コンテキスト アイテムのテキスト フィールドの動的なリンクをわかりやすい URL に変換するには：

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
string containsDynamicLinks = contextItem.Fields["text"].Value;
string containsFriendlyLinks = Sitecore.Links.LinkManager.ExpandDynamicLinks(
    containsDynamicLinks, Sitecore.Configuration.Settings.Rendering.SiteResolving);
string mediaPrefix = string.Empty;
foreach (string currentPrefix in
    Sitecore.Resources.Media.MediaManager.Provider.Config.MediaPrefixes)
{
    if (containsFriendlyLinks.IndexOf(currentPrefix, 0) > 0)
    {
        mediaPrefix = currentPrefix;
        break;
    }
}
if (mediaPrefix.Length > 0)
{
    finalMarkup = System.Text.RegularExpressions.Regex.Replace(containsFriendlyLinks,
        "([^\s/])" + mediaPrefix, "$1/" + mediaPrefix);
}
else
{
    finalMarkup = containsFriendlyLinks;
}
```

4.2 renderField パイプライン

renderField パイプラインを使って、フィールド値の動的なリンクをわかりやすい URL に変換し、ページ エディターのインライン編集時にサポートするフィールドタイプのためのインライン編集コントロールを生成することができます。たとえば、renderField パイプラインを使って、コンテキスト アイテムの Text フィールドをレンダリングするには：

```
Sitecore.Pipelines.RenderField.RenderFieldArgs args =  
    new Sitecore.Pipelines.RenderField.RenderFieldArgs();  
args.Item = Sitecore.Context.Item;  
args.FieldName = "text";  
Sitecore.Pipelines.CorePipeline.Run("renderField", args);  
string markup = args.Result.ToString();
```

renderField パイプラインは次のデータ テンプレート フィールド タイプをサポートします：

- Date（日付）
- Datetime（日時）
- Image（イメージ）
- Integer（整数）
- Multi-line Text（複数行テキスト）
- Number（数値）
- Rich Text（リッチ テキスト）
- Single-line Text（単一行テキスト）

フィールド値にアクセスするために renderField パイプラインを使うコンストラクトに関する情報は、「FieldRenderer Web コントロール」と「フィールド タイプ Web コントロール」のセクションを参照してください。

4.3 FieldRenderer Web コントロール

FieldRenderer Web コントロール (`Sitecore.Web.UI.WebControls.FieldRenderer`) を使って、フィールド値の動的なリンクをわかりやすい URL に変換し、ページ エディターのインライン編集時にサポートするフィールド タイプのためのインライン編集コントロールを生成することができます。FieldRenderer Web コントロールを Web コントロールとして使用できます。または静的な

`Sitecore.Web.UI.WebControls.FieldRenderer.Render()` メソッドを使用することができます。

FieldRenderer Web コントロールについての詳細は、『[コンテンツ API クックブック](#)』を参照してください。

FieldRenderer Web コントロールで使用される `renderField` パイプラインに関する詳細は、「renderField パイプライン」のセクションを参照してください。

FieldRenderer Web コントロールを Web コントロールとして使用するには、次のようなコントロールをレイアウトまたはサブレイアウトのファイルに追加します：

```
<sc:FieldRenderer runat="server" fieldname="//TODO: field" />
```

FieldRenderer Web コントロールの、または FieldRenderer Web コントロールから継承するフィールドタイプ Web コントロールの、`datasource` 属性を使って、フィールドを含むアイテムを指定することができます。（レンダリングのデフォルトのデータ ソース アイテムはコンテキスト アイテムです）

レンダリングにデータ ソースを指定する方法については、『[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。

静的な `Sitecore.Web.UI.WebControls.FieldRenderer.Render()` メソッドを起動して、フィールドの動的なリンクを変換し、ページ エディターのインライン編集時のインライン編集コントロールを生成することができます。たとえば、コンテキスト アイテムの `Title` という名前の単一行テキストフィールドのためのマークアップを生成するには：

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;  
string markup = Sitecore.Web.UI.WebControls.FieldRenderer.Render(contextItem,  
"title");
```

FieldRenderer Web コントロールの 1 つのインスタンスを別のコントロールのコントロール コレクションに追加することができます。次のサンプル コードのように、レイアウトまたはサブレイアウトのコードで `Page_Load()` メソッドを実装することができます。このサンプル コードではコンテキスト アイテムの `Title` フィールドの FieldRenderer Web コントロールを追加します。

```
protected void Page_Load(object sender, EventArgs e)  
{  
    Sitecore.Web.UI.WebControls.FieldRenderer titleRenderer =  
        new Sitecore.Web.UI.WebControls.FieldRenderer();  
    titleRenderer.FieldName = "title";  
    this.Controls.Add(titleRenderer);  
}
```

4.4 フィールド タイプ Web コントロール

このセクションに記載されているフィールド タイプ Web コントロールを使って、特定のタイプのフィールドの動的なリンクをわかりやすい URL に変換し、ページ エディターでサポートするフィールド タイプのためのインライン編集コントロールを生成することができます。フィールド タイプ Web コントロールは FieldRenderer Web コントロールから継承します。FieldRenderer Web コントロールに関する詳細は、「FieldRenderer Web コントロール」のセクションを参照してください。

FieldRenderer Web コントロールについての詳細は、『[コンテンツ API クックブック](#)』を参照してください。

4.4.1 リンク フィールド Web コントロール

リンク フィールド Web コントロール (`Sitecore.Web.UI.WebControls.Link`) を使って、General Link タイプのフィールドの値に基づいて、HTML アンカー要素 (“<a>”) を生成することができます。たとえば、リンク フィールド Web コントロールを使ってコンテキスト アイテムの GeneralLinkField という名前の General Link フィールドをレンダリングするには、下記のようなリンク フィールド Web コントロールをレイアウトまたはサブレイアウト ファイルに追加します：

```
<sc:link field="generallinkfield" runat="server" />
```

リンク フィールド Web コントロールの `text` 属性を使ってリンクのテキストを上書きすることができます。たとえば、リンク フィールド Web コントロールを使ってコンテキスト アイテムの GeneralLinkField という名前の General Link フィールドをレンダリングし、リンクのテキストを上書きするには：

```
<sc:link field="generallinkfield" runat="server" text="//TODO: replace with link text" />
```

または、テキストをリンク フィールド Web コントロールの中に入れて、リンクのテキストを上書きすることができます。

```
<sc:link field="generallinkfield" runat="server">
  //TODO: replace with link text
</sc:link>
```

メモ

リンク フィールド Web コントロールは、HTML イメージ要素や他の ASP.NET コントロールなどのマークアップ要素を含むことができます。

4.4.2 イメージ フィールド Web コントロール

イメージ フィールド Web コントロール (`Sitecore.Web.UI.WebControls.Image`) を使って、Image タイプのフィールドに基づいて、HTML イメージ要素 () を生成することができます。たとえば、イメージ フィールド Web コントロールを使ってコンテキスト アイテムの ImageField という名前の

Image フィールドを処理するには、下記のようなイメージ フィールド Web コントロールをレイアウトまたはサブレイアウト ファイルに追加します：

```
<sc:image field="imagefield" runat="server" />
```

追加の属性を使って追加のイメージ プロパティを指定することができます。たとえば、alt 属性とイメージ フィールド Web コントロールを使って、コンテキスト アイテムの ImageField という名前の Image フィールドの代替テキストを上書きするには：

```
<sc:image field="imagefield" runat="server" alt="//TODO: alternate text" />
```

4.4.3 日付フィールド Web コントロール

日付フィールド Web コントロール (Sitecore.Web.UI.WebControls.Date) を使って、Date または Datetime タイプのフィールドの文字列表現をレンダリングすることができます。たとえば、日付フィールド Web コントロールを使ってコンテキスト アイテムの DatetimeField という名前の [日付] フィールドをレンダリングするには、下記のような日付フィールド Web コントロールをレイアウトまたはサブレイアウト ファイルに追加します：

```
<sc:date field="datetimefield" runat="server" />
```

日付フィールド Web コントロールの format 属性を使って、.NET の日付と時間のフォーマット パターンを指定することができます。たとえば、日付フィールド Web コントロールを使ってコンテキスト アイテムの DatetimeField という名前の [日付] フィールドを、フルの日時と短い時間のパターンを使ってレンダリングするには：

```
<sc:date field="datetimefield" runat="server" format="f" />
```

.NET の日付と時間のフォーマット パターンに関する詳細は、[http://msdn.microsoft.com/en-us/library/97x6twsz\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/97x6twsz(VS.71).aspx) を参照してください。

4.4.4 テキスト フィールド Web コントロール

テキスト フィールド Web コントロール (Sitecore.Web.UI.WebControls.Text) を使って、複数行テキスト、リッチ テキスト、単一行テキストのタイプのフィールドをレンダリングすることができます。たとえば、テキスト フィールド Web コントロールを使ってコンテキスト アイテムの SingleLineTextField という名前の単一行テキスト フィールドをレンダリングするには、下記のようなテキスト フィールド Web コントロールをレイアウトまたはサブレイアウト ファイルに追加します：

```
<sc:text field="singlelinetextfield" runat="server" />
```

4.5 フィールドへのアクセスのサンプル

このセクションではいろいろなタイプのフィールドにアクセスするサンプル コードを提供します。

4.5.1 サンプル コード: 選択フィールドのアイテムへのリンク

次のサンプル コードを元にしてプレゼンテーション ロジックを実装することができます。このサンプル コードでは、コンテキスト アイテムの SelectionField という名前の Droplink、Droptree、Grouped Droplink タイプのフィールドで選択されたアイテムにリンクします。

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
Sitecore.Data.Fields.ReferenceField selectionField =
    contextItem.Fields["selectionfield"];

if (selectionField != null && selectionField.TargetItem != null)
{
    Sitecore.Data.Items.Item target = selectionField.TargetItem;
    string markup = String.Format(
        @"<a href=""{0}"">{1}</a>",
        Sitecore.Links.LinkManager.GetItemUrl(target),
        target.Name);
}
```

4.5.2 サンプル コード: ファイルまたはイメージ フィールドのメディアへのリンク

次のサンプル コードを元にしてプレゼンテーション ロジックを実装することができます。このサンプル コードでは、コンテキスト アイテムの FileField という名前の File タイプのフィールドで選択されたメディア アイテムにリンクします。:

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
Sitecore.Data.Fields.FileField fileField = contextItem.Fields["filefield"];

if (fileField != null && fileField.MediaItem != null)
{
    Sitecore.Data.Items.MediaItem media =
        new Sitecore.Data.Items.MediaItem(fileField.MediaItem);
    string markup = string.Format(
        @"<a href=""{0}"">{1}</a>",
        Sitecore.Resources.Media.MediaManager.GetMediaUrl(media),
        media.Name);
}
```

メモ

同様の方法を使って Image フィールドで選択されたイメージを処理することができます。

Sitecore.Data.Fields.FileField クラスに代えて Sitecore.Data.Fields.ImageField クラスを使用します。

4.5.3 サンプル コード: 複数選択フィールドのアイテムへのリンク

次のサンプル コードを元にしてプレゼンテーション ロジックを実装することができます。このサンプル コードでは、コンテキスト アイテムの MultiselectField という名前の Checklist、Multilist、Treelist、TreelistEx タイプのフィールドで選択されたアイテムにリンクします。

```
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
Sitecore.Data.Fields.MultilistField multiselectField =
    contextItem.Fields["multiselectfield"];

if (multiselectField != null && multiselectField.GetItems().Length>0)
{
    System.Text.StringBuilder markupBuilder = new System.Text.StringBuilder("<ul>");

    foreach (Sitecore.Data.Items.Item item in multiselectField.GetItems())
    {
        markupBuilder.Append(string.Format(
            @"<li><a href="{0}">{1}</a>",
            Sitecore.Links.LinkManager.GetItemUrl(item),
            item.Name));
    }

    markupBuilder.Append("</ul>");
    string markup = markupBuilder.ToString();
    output.Write(markup);
}
```

4.5.4 サンプル コード: ファイル ドロップ エリア フィールドのメディア アイテムへのリンク

このサンプル コードを使って [ファイル ドロップ エリア] フィールドのアイテムにリンクする Web コントロールを実装することができます。

```
namespace Sitecore.Sharedsource.Web.UI.WebControls
{
    using System;
    using System.Web.UI;

    public class FDAList : Sitecore.Web.UI.WebControl
    {
        public string FieldName
        {
            get;
            set;
        }

        protected override void DoRender(HtmlTextWriter output)
        {
            if (String.IsNullOrEmpty(this.FieldName)
                || Sitecore.Context.Item == null)
            {
                return;
            }

            Sitecore.Data.Fields.FileDropAreaField fdaField =
                Sitecore.Context.Item.Fields[this.FieldName];

            if (fdaField == null
                || String.IsNullOrEmpty(fdaField.Value)
                || fdaField.MediaFolder == null)
            {
                return;
            }

            Sitecore.Collections.ChildList children = fdaField.MediaFolder.Children;
```

```
        if (children.Count < 1)
        {
            return;
        }

        foreach (Sitecore.Data.Items.Item media in fdaField.MediaFolder.Children)
        {
            string url = Sitecore.StringUtil.EnsurePrefix(
                '/',
                Sitecore.Resources.Media.MediaManager.GetMediaUrl(media));
            string message = String.Format(
                @"<a href="{0}">{1}</a>",
                url,
                media.Name);
            output.WriteLine(message);
        }
    }
}
```

Chapter 5

言語の使用

この章では複数の言語によるアイテム データにアクセスするために使用する API についての情報を解説します。この章では Sitecore を使った言語の処理の概要を説明し、サンプル コードを提供します。

この章には次のセクションがあります。

- 言語の概要
- 言語を起動するサンプル コード

5.1 言語の概要

各アイテムは複数の言語を含むことができます。各言語は複数のバージョンを含むことができます。特に指定されていない限り、Sitecore はコンテキスト言語の各アイテムの最新のバージョンからフィールド値にアクセスします。コンテキスト言語に関する詳細は、「コンテキスト言語」のセクションを参照してください。

重要

アイテムがある言語のバージョン データを含んでいない場合、そのアイテムのすべてのバージョンのフィールド値は、その言語では空です。

メモ

共用フィールドの値はすべての言語のすべてのバージョンに適用されます。その言語のバージョン データのないアイテムは、その言語に共用フィールド値を含んでいます。

重要

コンテキスト言語にフィールド値が存在しないときに、Sitecore は代替言語に自動的にフォールバックしません。フォールバック言語をサポートするためのアドバイスについては、「フォールバック言語のサポート」のセクションを参照してください。

5.2 言語を起動するサンプル コード

このセクションでは言語を起動する API のサンプルコードを提供します。

5.2.1 サンプル コード: コンテキスト言語のバージョン データを持つアイテムへのリンク

`Sitecore.Data.Items.Item.Versions.Count` プロパティを使って、あるアイテムのある言語のバージョン データが存在するかどうかを知ることができます。次のサンプル コードを元にしてソリューションを実装することができます。このサンプル コードでは、コンテキスト言語でバージョン データを含み、コンテキスト デバイスでレイアウト詳細を持つ、コンテキスト アイテムの子のそれぞれへのリンクのリストを生成します。

```
System.Text.StringBuilder markup = new System.Text.StringBuilder();
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;

foreach (Sitecore.Data.Items.Item child in contextItem.Children)
{
    if (child.Versions.Count > 0
        && child.Visualization.GetLayout(Sitecore.Context.Device) != null)
    {
        string url = Sitecore.Links.LinkManager.GetItemUrl(child);
        markup.AppendFormat("<li><a href=\"\"{0}\"\">{1}</a></li>", url, child.Name);
    }
}

if (markup.Length > 0)
{
    string list = "<ul>" + markup.ToString() + "</ul>";
}
```

5.2.2 サンプル コード: 言語国旗 Web コントロール

次のサンプル コードを元にして Web コントロールを実装することができます。このサンプル コードでは、コンテキスト アイテムの代替言語に対応する国旗をレンダリングします。

```
using System;

namespace Sitecore.Sharedsource.Web.UI.WebControls
{
    public class LanguageFlags : Sitecore.Web.UI.WebControl
    {
        protected override void DoRender(System.Web.UI.HtmlTextWriter output)
        {
            string spacer = " ";
            System.Text.StringBuilder flags = new System.Text.StringBuilder();

            foreach (Sitecore.Globalization.Language language in
                Sitecore.Context.Database.Languages)
            {
                if (language == Sitecore.Context.Language)
                {
                    flags.Append(this.GetLanguageFlag(language) + spacer);
                }
                else
                {
                    Sitecore.Data.Items.Item langItem =
                        Sitecore.Context.Database.GetItem(Sitecore.Context.Item.ID, language);

                    if (langItem.Versions.Count > 0)
                    {

```

```

        string markup = String.Format(
            @"<a href="{0}">{1}</a>{2}",
            this.GetLanguageUrl(langItem),
            this.GetLanguageFlag(language),
            spacer);
        flags.Append(markup);
    }
}

if (flags.Length > 0)
{
    flags.Length = flags.Length - spacer.Length;
    output.Write(flags);
}

private string GetLanguageFlag(Sitecore.Globalization.Language language)
{
    string icon = language.GetIcon(Sitecore.Context.Database);

    if (icon.StartsWith("~"))
    {
        icon = Sitecore.StringUtil.EnsurePrefix('/', icon);
    }
    else if ((!icon.StartsWith("/") && (!icon.Contains(":"))))
    {
        icon = Sitecore.Resources.Images.GetThemedImageSource(icon);
    }

    return String.Format(
        @"",
        icon,
        language.GetDisplayName());
}

private string GetLanguageUrl(Sitecore.Data.Items.Item item)
{
    Sitecore.Links.UrlOptions urlOptions =
        Sitecore.Links.LinkManager.GetDefaultUrlOptions();
    urlOptions.Language = item.Language;
    urlOptions.LanguageEmbedding = Sitecore.Links.LanguageEmbedding.Always;
    return Sitecore.Links.LinkManager.GetItemUrl(item, urlOptions);
}
}

```

言語国旗 Web コントロールはコンテキスト データベースの各言語に繰り返して適用されます。その言語がコンテキスト言語と同じである場合、コントロールはその言語に関連付けられた国旗を表示します。または、その言語でコンテキスト アイテムのバージョンが存在する場合、コントロールはその言語に関連付けられた国旗をその言語のコンテキスト アイテムへのリンクとして表示します。

Web コントロールの実装についての詳細は、『[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。

5.2.3 サンプル コード: ユーザー プロファイルからコンテキスト言語を知る

次のサンプル コードを元にして要求処理パイプラインを実装することができます。このサンプル コードでは、cookie、クエリ文字列、URL パスが言語を指定していない場合に、コンテキスト ユーザーのプロファイルからコンテキスト言語を知ることができます。

```

namespace Sitecore.Sharedsource.Pipelines.HttpRequest
{
    public class ProfileLanguageResolver :
        Sitecore.Sharedsource.Pipelines.HttpRequest.LanguageProcessor
    {
    }
}

```



```

{
    protected override bool IsLanguageKnown()
    {
        return this.QueryHasLanguage()
            || this.PathHasLanguage()
            || this.CookieHasLanguage();
    }

    public void Process(Sitecore.Pipelines.HttpRequest.HttpRequestArgs args)
    {
        if ((!this.IsLanguageKnown()) && this.ProfileHasLanguage())
        {
            Sitecore.Globalization.Language language;

            if (Sitecore.Globalization.Language.TryParse(
                Sitecore.Context.User.Profile.RegionalIsoCode, out language))
            {
                Sitecore.Context.SetLanguage(language, true);
                Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;

                if (contextItem != null)
                {
                    Sitecore.Context.Item =
                        Sitecore.Context.Database.GetItem(contextItem.ID, language);
                }
            }
        }
    }
}

```

Insert the processor after the item resolver within the

/configuration/sitecore/pipelines/httpRequestBegin element in web.config:

```

<httpRequestBegin>
...
  <processor type="Sitecore.Pipelines.HttpRequest.ItemResolver, Sitecore.Kernel" />
  <processor type=
    "Sitecore.Sharedsource.Pipelines.HttpRequest.ProfileLanguageResolver, Assembly" />
...

```

The

Sitecore.Sharedsource.Pipelines.HttpRequest.ProfileLanguageResolver.Process() method sets the context language to the language specified in the user's profile if the user is not anonymous and has a profile language.

言語プロセッサ ベース クラス

言語プロセッサ ベース クラス

Sitecore.Sharedsource.Pipelines.HttpRequest.LanguageProcessor は

QueryHasLanguage()、PathHasLanguage()、CookieHasLanguage()、ProfileHasLanguage() のメソッドを提供します。これらは HTTP 要求のどのプロパティが言語情報を含むかを指定します。このベース クラスから継承するプロセッサ クラスは IsLanguageKnown() メソッドを上書きする必要があります。これは高い優先度の基準がコンテキスト言語を決めるかどうかを指定します。

```

using System;

namespace Sitecore.Sharedsource.Pipelines.HttpRequest
{
    public abstract class LanguageProcessor
    {
        protected bool QueryHasLanguage()
        {
            return !String.IsNullOrEmpty(Sitecore.Web.WebUtil.GetQueryString("sc lang"));
        }
    }
}

```

```
protected bool PathHasLanguage()
{
    return Sitecore.Context.Data.FilePathLanguage != null;
}

protected bool CookieHasLanguage()
{
    if (Sitecore.Context.Site != null)
    {
        string cookie = Sitecore.Web.WebUtil.GetCookieValue(
            Sitecore.Context.Site.GetCookieKey("lang"));

        if (!String.IsNullOrEmpty(cookie))
        {
            return true;
        }
    }

    return false;
}

protected bool ProfileHasLanguage()
{
    return Sitecore.Context.IsLoggedIn &&
        !String.IsNullOrEmpty(Sitecore.Context.User.Profile.RegionalIsoCode);
}

protected abstract bool IsLanguageKnown();
}
```

メモ

言語の解決とアイテムの解決の間のプロセッサがコンテキスト言語に依存している場合、カスタムの言語プロセッサを言語の解決の直後に挿入します。または、アイテムの解決の後にカスタムのプロセッサを挿入した場合、そのプロセッサは `Sitecore.Data.Items.Items.Versions.Count` プロパティを使って、どの言語が各言語のバージョン データを含んでいるかを知ることができます。「サンプル コード: スレッド カルチャーの設定」のセクションに記載されているように、カルチャーの解決の前にプロセッサを挿入します。

5.2.4 サンプル コード: Web クライアントの好みから言語を決定する

次のサンプル コードを元にして要求処理パイプラインを実装することができます。このサンプル コードでは Web クライアントの好みからコンテキスト言語を決定します。

```
using System;

namespace Sitecore.Sharedsource.Pipelines.HttpRequest
{
    public class ClientLanguageResolver : LanguageProcessor
    {
        protected override bool IsLanguageKnown()
        {
            return QueryHasLanguage() || PathHasLanguage() || CookieHasLanguage() ||
                ProfileHasLanguage();
        }

        public void Process(Sitecore.Pipelines.HttpRequest.HttpRequestArgs args)
        {
            if (!this.IsLanguageKnown())
            {
                string langs = args.Context.Request.ServerVariables["HTTP_ACCEPT_LANGUAGE"];

                if (!String.IsNullOrEmpty(langs))
                {

```

```
foreach (string requested in Sitecore.StringUtil.Split(langs, ',', true))
{
    Sitecore.Globalization.Language language;
    string languageCode = requested;

    if (languageCode.IndexOf(';') > -1)
    {
        languageCode = languageCode.Substring(0, languageCode.IndexOf(';'));
    }

    Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;

    if (Sitecore.Globalization.Language.TryParse(languageCode, out language)
        && (contextItem == null
            || contextItem.Versions.Count > 0))
    {
        Sitecore.Context.SetLanguage(language, true);

        if (contextItem != null)
        {
            Sitecore.Context.Item =
                Sitecore.Context.Database.GetItem(contextItem.ID, language);
        }

        return;
    }
}
}
```

web.config の /configuration/sitecore/pipelines/httpRequestBegin 要素の中で言語の解決の後にプロセッサを挿入します。

```
<httpRequestBegin>
...
  <processor type="Sitecore.Pipelines.HttpRequest.LanguageResolver, Sitecore.Kernel"
/>

  <processor type=
    "Sitecore.Sharedsource.Pipelines.HttpRequest.ClientLanguageResolver, Assembly" />
...

```

Sitecore.Sharedsource.Pipelines.HttpRequest.ClientLanguageResolver.Process () メソッドは Web クライアントから HTTP 要求で送信された HTTP_ACCEPT_LANGUAGE ヘッダー変数を解析します。コンテキスト アイテムが不明な場合またコンテキスト アイテムがバージョン データを含む最初の言語である場合、このメソッドは最初に指定された言語をコンテキスト言語に設定します。ベース クラス Sitecore.Sharedsource.Pipelines.HttpRequest.LanguageProcessor に関する詳細は、「言語プロセッサ ベース クラス」のセクションを参照してください。

メモ

「サンプル コード: ユーザー プロファイルからコンテキスト言語を知る」のセクションで記載されたようにユーザー プロファイルからコンテキスト言語を設定しなかった場合、`ProfileHasLanguage()` メソッドを削除し、`IsLanguageKnown()` メソッドを適宜に更新します。

メモ

前のセクション「言語プロセッサー ベース クラス」でのコンテキスト言語に関するメモを参照してください。

5.2.5 サンプル コード: スレッド カルチャーの設定

ASP.NET はコンテキスト言語に関連付けられた地域コードに従って日付と数値を自動的に整形しません。次のサンプル コードを元にしてカスタムのプロセッサを実装することができます。このサンプル コードではスレッドにカルチャーを設定し、日付と数値のフォーマットを制御します。

```
namespace Sitecore.Sharedsource.Pipelines.HttpRequest
{
    public class CultureResolver
    {
        public void Process(Sitecore.Pipelines.HttpRequest.HttpRequestArgs args)
        {
            if (!Sitecore.Context.Language.CultureInfo.IsNeutralCulture)
            {
                System.Threading.Thread.CurrentThread.CurrentUICulture =
                    Sitecore.Context.Language.CultureInfo;
                System.Threading.Thread.CurrentThread.CurrentCulture =
                    Sitecore.Context.Language.CultureInfo;
            }
        }
    }
}
```

web.config の /configuration/sitecore/pipelines/httpRequestBegin 要素の中で言語の解決の後にカルチャーの解決を挿入します。

```
<httpRequestBegin>
...
<processor type="Sitecore.Pipelines.HttpRequest.LanguageResolver, Sitecore.Kernel"
/>
<processor
type="Sitecore.Sharedsource.Pipelines.HttpRequest.CultureResolver, Assembly" />
...
```

メモ

ASP.NET は HTTP 要求を処理した後でスレッドのカルチャーを自動的にリセットします。

メモ

セクション「サンプル コード: ユーザー プロファイルからコンテキスト言語を知る」、「サンプル コード: Web クライアントの好みから言語を決定する」、「サンプル コード: アイテムのフォールバック言語」に記載されているように、カスタムの言語の解決の後にカルチャーの解決を挿入します。

5.2.6 フォールバック言語のサポート

次の手順を使ってフォールバック言語をサポートすることができます。これによってフィールドまたはアイテムがコンテキスト言語の値を持っていない場合に、代替言語から値を表示することができます。

1. テンプレート マネージャーまたはコンテンツ エディターで [Custom Language] という名前のカスタム データ テンプレートを作成します。

データ テンプレートを保守する方法については、『[データ定義クックブック](#)』を参照してください。

2. **[Custom Language]** データ テンプレートに **[Custom]** という名前のセクションを追加します。
3. **[Custom Language]** データ テンプレートの **[Custom]** セクションに **[Fallback Language]** という名前の **[Droplist]** フィールドを追加します。
4. **[Custom Language]** データ テンプレートの **[Custom]** セクションの **[Fallback Language]** フィールドで **[Shared]** を選択します。
5. **[Custom Language]** データ テンプレートの **[Custom]** セクションの **[Fallback Language]** フィールドの **[Source]** プロパティを `/sitecore/system/languages` に設定します。
6. `System/Language` データ テンプレートのベース テンプレートを **Custom Language** テンプレートを含むように構成します。
7. **コンテンツ エディター**で `/Sitecore/System/Languages` アイテムの下で、各言語定義アイテムの **[Custom]** セクションの **[Fallback Language]** フィールドで、フォールバック言語を選択します。
8. Visual Studio の Web アプリケーション プロジェクトでは
`Sitecore.Globalization.Language` クラスの拡張メソッドを実装し、ある言語に対するフォールバック言語へのアクセスを提供します。

```
using System;

namespace Sitecore.Sharedsource.Globalization
{
    public static class LanguageExtensions
    {
        private const string FALLBACK_LANGUAGE_FIELD = "fallback language";

        public static Sitecore.Globalization.Language GetFallbackLanguage(
            this Sitecore.Globalization.Language language)
        {
            if (Sitecore.Context.Database != null)
            {
                Sitecore.Data.ID languageId =
                    Sitecore.Data.Managers.LanguageManager.GetLanguageItemId(
                        language,
                        Sitecore.Context.Database);

                if (languageId != (Sitecore.Data.ID) null)
                {
                    Sitecore.Data.Items.Item languageItem =
                        Sitecore.Context.Database.GetItem(languageId);

                    if (languageItem != null)
                    {
                        string fallbackLanguageName = languageItem[FALLBACK_LANGUAGE_FIELD];

                        if (!String.IsNullOrEmpty(fallbackLanguageName))
                        {
                            return Sitecore.Globalization.Language.Parse(fallbackLanguageName);
                        }
                    }
                }
            }
        }
    }
}
```

```
    }  
  }  
  return null;  
}  
}
```

メモ

この拡張機能を使用するコードは、明示的に `Sitecore.Sharedsource.Globalization` 名前空間を参照する必要があります。

```
using Sitecore.Sharedsource.Globalization;
```

`Sitecore.Globalization.Language.GetFallbackLanguage()` 拡張メソッドを使って、言語に関連付けられたフォールバック言語にアクセスすることができます。たとえば、コンテキスト言語のフォールバック言語にアクセスするには：

```
using Sitecore.Sharedsource.Globalization;  
...  
Sitecore.Globalization.Language contextLanguage = Sitecore.Context.Language;  
Sitecore.Globalization.Language fallbackLanguage =  
    contextLanguage.GetFallbackLanguage();
```

メモ

あるアイテムは、各フィールドの値を含むことなく、ある言語のバージョンを含むことができます。

サンプル コード: フィールドのフォールバック言語

あるフィールドがコンテキスト言語の値を持っていない場合、フォールバック言語から対応するフィールド値を読み出すことができます。フォールバック言語をサポートするための方法については、「フォールバック言語のサポート」のセクションを参照してください。次のサンプル コードを元にしてロジックを実装することができます。このサンプル コードではコンテキスト言語、またはそのフィールドの値を持つ最初のフォールバック言語のコンテキスト アイテムから、Title フィールドの値を読み出します。

```
using Sitecore.Sharedsource.Globalization;  
...  
Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;  
Sitecore.Data.Fields.Field title = contextItem.Fields["title"];  
string markup = String.Empty;  
  
if (!String.IsNullOrEmpty(title.Value) || Sitecore.Context.PageMode.IsPageEditor)  
{  
    markup = Sitecore.Web.UI.WebControls.FieldRenderer.Render(contextItem, "title");  
}  
else  
{  
    Sitecore.Globalization.Language language = Sitecore.Context.Language;  
  
    do  
    {  
        language = language.GetFallbackLanguage();  
  
        if (language != null)  
        {
```

```
        Sitecore.Data.Items.Item languageItem =
            contextItem.Database.GetItem(contextItem.ID, language);
        title = languageItem.Fields[title.ID];
    }
}
while (language != null && String.IsNullOrEmpty(title.Value));

if (!String.IsNullOrEmpty(title.Value))
{
    markup = title.Value;
}
}
```

重要

ページ エディター内を除き、コンテキスト言語以外の言語のフィールド値にはインライン編集コントロールを提供しないでください。

サンプル コード: アイテムのフォールバック言語

コンテキスト アイテムがコンテキスト言語のバージョン データを含んでいない場合、コンテキスト言語をフォールバック言語に設定することができます。フォールバック言語をサポートするための方法については、「フォールバック言語のサポート」のセクションを参照してください。次のサンプル コードを元にして要求処理パイプラインを実装することができます。このサンプル コードでは、コンテキスト アイテムがコンテキスト言語のバージョン データを含んでいない場合に、コンテキスト言語をフォールバック言語に設定します。

```
public class FallbackLanguageProcessor
{
    public void Process(Sitecore.Pipelines.HttpRequest.HttpRequestArgs args)
    {
        Sitecore.Data.Items.Item contextItem = Sitecore.Context.Item;
        if (contextItem == null || contextItem.Versions.Count > 0)
        {
            return;
        }
        Sitecore.Globalization.Language contextLanguage =
            Sitecore.Context.Language;

        Sitecore.Globalization.Language fallbackLanguage =
            contextLanguage.GetFallbackLanguage();
        if (fallbackLanguage == null)
        {
            return;
        }
        if (fallbackLanguage.Name.Equals(contextLanguage.Name,
            StringComparison.InvariantCultureIgnoreCase))
        {
            return;
        }

        Sitecore.Data.Database contextDatabase = Sitecore.Context.Database;
        Sitecore.Data.Items.Item fallbackItem =
            contextDatabase.GetItem(Sitecore.Context.Item.ID, fallbackLanguage);
        if (fallbackItem != null && fallbackItem.Versions.Count > 0)
        {
            Sitecore.Context.Item = fallbackItem;
            Sitecore.Context.Language = fallbackLanguage;
        }
    }
}
```

メモ

このサンプル コードは、Sitecore でフォールバック言語を実装するための非常に簡単な方法ですが、複雑なソリューションにおいては十分ではないことがあります。詳細は、共有ソース ライブラリの「言語フォールバック アイテム プロバイダー」モジュールを参照してください。

web.config の httpRequestBegin パイプラインで、アイテムの解決後にフォールバック言語の解決を挿入します。

```
<httpRequestBegin>
...
  <processor type="Sitecore.Pipelines.HttpRequest.ItemResolver, Sitecore.Kernel" />
  <processor type=
    "Sitecore.Sharedsource.Pipelines.HttpRequest.FallbackLanguageProcessor, Assembly"
  />
/>
...
```

コンテキスト アイテムが不明な場合、または存在しない場合、またはコンテキスト ユーザーによる読み取りアクセスを禁止するアクセス権を持つ場合、コンテキスト アイテムがコンテキスト言語のバージョン データを含む場合、フォールバック言語プロセッサは何もしません。または、コンテキスト言語にフォールバック言語が存在しアイテムがその言語のバージョン データを含む場合、その言語をコンテキスト アイテムに設定し、コンテキスト アイテムをその言語に更新します。

メモ

セクション「サンプル コード: ユーザー プロファイルからコンテキスト言語を知る」、「サンプル コード: Web クライアントの好みから言語を決定する」に記載されているように、他のカスタムの言語の解決の後にフォールバック言語の解決を挿入します。

Chapter 6

エラー処理

この章ではプレゼンテーション コンポーネントで発生するエラーと例外に対応するために使用する情報を説明します。この章ではまず例外処理の概要を説明し、次に C# の `try/catch` ブロックと Sitecore のエラーページについて説明します。続いてレンダリング、レイアウト、アプリケーションで例外を捕捉する方法を説明します。

この章には次のセクションがあります。

- 例外処理の概要
- Try/Catch ブロック
- Sitecore のエラーページ
- XSL レンダリングの例外の捕捉方法
- Web コントロール レンダリングの例外の捕捉方法
- レイアウトの例外の捕捉方法
- アプリケーションの例外の捕捉方法

6.1 例外処理の概要

例外とはメソッドがタスクの処理を完了できずに現行のコードブロックを終了させるような特定のタイプのエラー状態による事象を指します。例外はそのタイプの例外をハンドルするレベルに達するまでコール スタックに上ります。ASP.NET はプレゼンテーション コンポーネントにスローされた例外を処理するために使用できる追加機能を提供しています。

発生したすべての例外の自動ロギングに加え、Sitecore は XSL レンダリングと Web コントロールの構文エラーと例外をハンドルする機能を提供しています。さらに特定のエラー状態についてのカスタム エラー ページを提供しています。

XSL レンダリングが構文エラーを含むか、または例外を発生させる XSL 拡張コントロールまたはメソッドを呼び出した場合、Sitecore は XSL レンダリングとエラーに関する情報のログを取り、それを表示します。他のプレゼンテーション コンポーネントで例外が発生した場合、Sitecore はその例外のログを取り、ASP.NET が正常に例外を処理します。

ASP.NET の例外処理に関する詳細は、以下を参照してください。

- [http://msdn.microsoft.com/en-us/library/ms229014\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/ms229014(VS.80).aspx)
- <http://msdn.microsoft.com/en-us/library/ms954830.aspx>
- <http://msdn.microsoft.com/en-us/library/aa479319.aspx>
- <http://msdn.microsoft.com/en-us/library/w16865z6.aspx>

6.1.1 ASP.NET の例外処理

ASP.NET は `web.config` の `/configuration/system.web/customErrors` 要素に従って例外を処理します。

`web.config` の `/configuration/system.web/customErrors` 要素の `mode` 属性の値が `RemoteOnly` である場合、ASP.NET は Web サーバー上で発生した HTTP 要求の例外のすべての詳細を返しますが、すべての Web クライアントには一般的なエラーメッセージを返すかリダイレクトします。

`web.config` の `/configuration/system.web/customErrors` 要素の `mode` 属性の値が `Off` である場合、ASP.NET は例外メッセージとすべてのスタックのトレースを Web クライアントに HTML で返します。これらの例外の詳細を公開することはセキュリティ リスクを生じさせる可能性があります。

`web.config` の `/configuration/system.web/customErrors` 要素の `mode` 属性の値が `On` である場合、ASP.NET は Web クライアントを `statusCode` に 500 を含む `<error>` 要素の `redirect` 属性で指定された URL にリダイレクトします。たとえば、次の構成では、プレゼンテーション コンポーネントが例外をスローした場合、ASP.NET は Web クライアントを `/errors/500.htm` にリダイレクトします：

```
<customErrors mode="On" defaultRedirect="/errors/generic.htm">
  <error statusCode="500" redirect="/errors/500.htm" />
</customErrors>
```

```
</customErrors>
```

メモ

エラーが生じたページを示すために、ASP.NET は `aspxerrorpath` クエリ文字列パラメーターを URL に追加します。

メモ

`<error>` 要素を追加して、別のエラー状態の場合は Web クライアントを別のページにリダイレクトすることができます。

`statusCode` が 500 である `<error>` 要素が存在しない場合、ASP.NET は Web クライアントを `customErrors` 要素の `defaultRedirect` 属性で指定された URL にリダイレクトします。たとえば、次の構成では、プレゼンテーション コンポーネントが例外をスローした場合、ASP.NET は Web クライアントを `/errors/generic.htm` にリダイレクトします：

```
<customErrors mode="On" defaultRedirect="/errors/generic.htm" />
```

メモ

エラーが生じたページを示すために、ASP.NET は `aspxerrorpath` クエリ文字列パラメーターを URL に追加します。

`defaultRedirect` 属性が `<customErrors>` 要素に存在しない場合は、ASP.NET は一般的な HTML エラーメッセージを Web クライアントに返します。

ヒント

例外処理のロジックをテストするには、`web.config` の `/configuration/system.web/customErrors` 要素の `mode` 属性を `On` に設定します。または、`mode` 属性を `Off` または `RemoteOnly` に設定します。

6.2 Try/Catch ブロック

C# の `try/catch` ブロックを使ってローカルで例外を捕捉することができます。たとえば、次のコードは除算の試みからスローされた例外のログを取ります。

```
int numerator = 5;
int denominator = 0;

try
{
    int quotient = numerator/denominator;

    //TODO: additional logic
}
catch (Exception exception)
{
    string message = String.Format("Unable to divide {0} by {1}", numerator,
denominator);
    Sitecore.Diagnostics.Log.Error(message, exception, this);
}

//TODO: additional logic
```

6.3 Sitecore のエラーページ

`web.config` の `/configuration/sitecore/settings/setting` 要素の `value` 属性は、次の表で指定される各種のエラー状態の下で Sitecore が起動する URL を指定します。

設定	エラー状態
ErrorPage	HTTP 要求が特定のエラー ハンドラーのない既知のエラー状態を生じた。
ItemNotFound	URL がデータベースのアイテムに対応しない。
LayoutNotFoundUrl	コンテキスト アイテムがコンテキスト デバイスに対する有効なレイアウト詳細を持っていない。
NoAccessUrl	コンテキスト ユーザーがコンテキスト アイテムの読み取りアクセス権を持っていない。
NoLicenseUrl	Sitecore が正しくライセンスされていない

メモ

`Sitecore.Configuration.Settings` クラスはこれらの設定に対応するプロパティを公開します。

メモ

構成によっては、ASP.NET は必ずしも IIS に受信されたすべての HTTP 要求を処理するとは限りません。Sitecore は ASP.NET に処理されたすべての HTTP 要求は処理しません。一貫性を保つため、`web.config` の Sitecore のエラー ページ設定と、`web.config` の `/configuration/system.web/customErrors` 要素の中の ASP.NET エラーページと、IIS 管理コンソールの IIS エラーページを、同じ値に設定することができます。

6.4 XSL レンダリングの例外の捕捉方法

Sitecore は XSL レンダリングから生じたすべての例外のログを取り、クリアし、例外に関する情報を Web クライアントに送信されるコンテンツに埋め込みます。

XSL レンダリングはコンパイル時におけるエラー検出をサポートしません。Sitecore は各 XSL レンダリングを、アプリケーション再起動後にシステムが最初にそれを起動した時にコンパイルします。レイアウト エンジンが実行時に XSL レンダリングで構文エラーを生じた場合、Sitecore はレンダリングが出力を生成する前に例外を発生させます。

XSL 拡張コントロールとメソッドは例外をスローできます。構文エラーを含まない XSL レンダリングは、例外を生じる XSL 拡張メソッドまたはコントロールが起動する前に、出力を生成することができます。

エラー コントロールを使って XSL レンダリングの例外と構文エラーを捕捉することができます。XSL レンダリングでレイアウト エンジンが構文エラーまたは例外を生じた場合、レイアウト エンジンは

/app_config/prototypes.config ファイルの /configuration/ErrorControl 要素の type 属性で指定されたコントロールを起動します。デフォルトの Sitecore.Web.UI.WebControls.StandardErrorControl がエラーに関する情報をレンダリングします。

メモ

例外の前にレンダリングに生成された出力は、エラー コントロールの出力よりも前に表示されます。

カスタムのエラー コントロールを実装するには、Sitecore.Web.UI.WebControls.ErrorControl から継承するクラスを作成し、Clone() メソッドを実装します。ベース クラスの Message と Details プロパティを使って、エラーメッセージとスタック トレースにアクセスします。たとえば、次のサンプル コードを元にしてエラー コントロールを実装することができます。このサンプル コードでは、XSL レンダリングが構文エラーを含むか、または例外をスローする .NET XSL 拡張コントロールまたはメソッドを起動した場合、web.config の /configuration/system.web/customErrors 要素の mode 属性によって、Web クライアントをわかりやすいエラー ページにリダイレクトするか、またはエラーに関する情報をレンダリングします：

```
using System;
using System.Web.UI;

namespace Sitecore.Sharedsource.Web.UI.WebControls
{
    public class ErrorControl : Sitecore.Web.UI.WebControls.StandardErrorControl
    {
        public override Sitecore.Web.UI.WebControls.ErrorControl Clone()
        {
            return (Sitecore.Web.UI.WebControls.ErrorControl) MemberwiseClone();
        }

        protected override void DoRender(HtmlTextWriter output)
        {
            Sitecore.Diagnostics.Log.Error(this.Message, this);
            ErrorHandler helper = new ErrorHandler();
            if (helper.ShouldRedirect())
            {
                helper.Redirect();
            }
            else
            {
                // Render error message and stack trace
            }
        }
    }
}
```

```
        {  
            base.DoRender(output);  
        }  
    }  
}
```

エラー コントロールの `DoRender()` メソッドがエラーのログを取り、次のヘルパー クラスを使って、Web クライアントをリダイレクトするか、またはベース クラスの `DoRender()` メソッドを呼び出してエラーに関する詳細をレンダリングするか、を決定します。

```
using System;  
using System.Configuration;  
using System.Web.Configuration;  
  
namespace Sitecore.Sharedsource.Web.UI  
{  
    public class ErrorHandler  
    {  
        private CustomErrorsSection _config = null;  
        private string error500url = null;  
  
        public CustomErrorsSection Config  
        {  
            get  
            {  
                if ( _config == null )  
                {  
                    Configuration config =  
                        WebConfigurationManager.OpenWebConfiguration("/");  
                    this._config =  
                        (CustomErrorsSection) config.GetSection("system.web/customErrors");  
                    Sitecore.Diagnostics.Assert.IsNotNull(this._config, "customErrors");  
                }  
  
                return this._config;  
            }  
        }  
  
        public string Error500Url  
        {  
            get  
            {  
                if (this.error500url == null && this.Config != null)  
                {  
                    CustomError error500 =  
                        this.Config.Errors.Get(500.ToString(  
                            System.Globalization.CultureInfo.InvariantCulture));  
  
                    if (error500 != null && !String.IsNullOrEmpty(error500.Redirect))  
                    {  
                        this.error500url = error500.Redirect;  
                    }  
  
                    if (String.IsNullOrEmpty(this.error500url))  
                    {  
                        this.error500url = Config.DefaultRedirect;  
                    }  
  
                    if (this._error500url == null)  
                    {  
                        this.error500url = String.Empty;  
                    }  
                }  
  
                return this._error500url;  
            }  
        }  
  
        public bool ShouldRedirect()  
        {  

```

```
        if (this.Config == null
            || this.Config.Mode == System.Web.Configuration.CustomErrorsMode.On
            || (this.Config.Mode ==
                System.Web.Configuration.CustomErrorsMode.RemoteOnly
                && !System.Web.HttpContext.Current.Request.IsLocal))
        {
            return true;
        }

        return false;
    }

    public void Redirect()
    {
        if (!String.IsNullOrEmpty(Error500Url))
        {
            Sitecore.Web.WebUtil.Redirect(
                this.Error500Url + "?aspxerrorpath="
                + Sitecore.Web.WebUtil.GetLocalPath(Sitecore.Context.RawUrl));
        }
        else
        {
            Sitecore.Web.WebUtil.RedirectToErrorPage(
                "//TODO: replace with friendly error message");
        }
    }
}
```

Web コントロールの実装についての詳細は、『[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

`Sitecore.Web.WebUtil.RedirectToErrorPage()` メソッドは Web クライアントを `web.config` の `/configuration/sitecore/settings/setting` 要素の `name ErrorPage` である `value` 属性で指定されたページにリダイレクトし、パラメーターで指定されたメッセージを `error` という名前の URL パラメーターとして渡します。

6.4.1 エラー ヘルパー

`Sitecore.Sharedsource.Web.UI.ErrorHelper` クラスは Web クライアントをわかりやすいエラー ページへリダイレクトするためのメソッドを提供します。`web.config` の `/configuration/system.web/customErrors` セクションの `mode` 属性が `On` である場合、または `RemoteOnly` であり現行の HTTP 要求がローカルでない場合、`ShouldRedirect()` メソッドは `True` を返します。`Redirect()` メソッドは Web クライアントを、次のリストから最初に定義された値にリダイレクトし、適切なクエリ文字列パラメーターを追加します：

- `web.config` の `/configuration/system.web/customErrors/error` 要素で `statusCode` が 500 で指定されたページ
- `web.config` の `/configuration/system.web/customErrors` 要素の `defaultRedirect` 属性
- `web.config` の `/configuration/sitecore/settings/setting` 要素の `name ErrorPage` である `value` 属性

メモ

エラー コントロールを Sitecore に登録する必要はありません。

6.5 Web コントロール レンダリングの例外の捕捉方法

「例外処理の概要」のセクションに記載されているように、Sitecore は、ASP.NET が Web コントロールを処理する前に、Web コントロールがスローした例外のログを取ります。次のサンプル コードを元にして Web コントロールを実装することができます。このサンプル コードでは「XSL レンダリングの例外の捕捉方法」のセクションに記載されているように、エラー コントロールを使って例外を捕捉します。

```
using System;
using System.Web.UI;

namespace Sitecore.Sharedsource.Web.UI.WebControls
{
    public class ExceptionThrower : Sitecore.Web.UI.WebControl
    {
        protected override void DoRender(HtmlTextWriter output)
        {
            try
            {
                throw new System.NotImplementedException();
            }
            catch (Exception ex)
            {
                Sitecore.Diagnostics.Log.Error(
                    "Web control exception in " + GetType().ToString(), ex, this);
                RenderError(ex.Message, ex.ToString(), output);
            }
        }
    }
}
```

Web コントロールの `DoRender()` メソッドはすべての例外のログを取り、それからさらに行うエラー処理のために、「XSL レンダリングの例外の捕捉方法」のセクションに記載されているように、エラー コントロールを起動します。

6.6 レイアウトの例外の捕捉方法

次のサンプル コードを元にしてソリューションを実装することができます。このサンプル コードではレイアウトのコード ファイルで `Page_Error()` メソッドを使って下位レベルで捕捉できない例外を捕捉し、わかりやすいエラー ページへリダイレクトします：

```
private void Page_Error(object sender, EventArgs e)
{
    string message = String.Format(
        "Layout exception in {0} processing {1}",
        GetType().ToString(),
        Sitecore.Context.RawUrl);
    Sitecore.Diagnostics.Log.Error(message, Server.GetLastError(), this);
    Sitecore.Sharedsource.Web.UI.ErrorHelper helper =
        new Sitecore.Sharedsource.Web.UI.ErrorHelper();

    if (helper.ShouldRedirect())
    {
        helper.Redirect();
    }
    else
    {
        Server.ClearError();
    }
}
```

`Page_Error()` メソッドが例外のログを取り、それからさらに行うエラー処理のために、「エラー ヘルパー」のセクションで記載されているように、エラーヘルパーを使います。

6.7 アプリケーションの例外の捕捉方法

ASP.NET アプリケーションファイル `global.asax` の `Application_Error()` メソッドを使って、下位レベルで捕捉できないすべての例外を捕捉することができます。必要なコードを `global.asax` にインラインで配置するか、または `global.asax` のためのコード ファイルを実装することができます。次のサンプルコードを元にしてコード ファイルを使ったアプリケーションでの例外処理を実装することができます。このサンプルコードでは Web クライアントをわかりやすいエラー ページにリダイレクトします。

1. Visual Studio の Web アプリケーション プロジェクトの Visual Studio ソリューション エクスプローラーで、プロジェクトまたは適当なディレクトリーで右クリックし、次に [追加] をクリックし、[クラス] をクリックします。[新しい項目の追加] ダイアログが表示されます。
2. [新しい項目の追加] ダイアログ ボックスで [カテゴリー] ツリーで [Visual C#] を選択します。
3. [テンプレート] リストで [クラス] を選択します。
4. [名前] フィールドで [Global.cs] と入力し、[追加] をクリックします。
5. `Global.cs` の内容を下記で置換し、名前空間とクラスの名前を適宜に更新します。

```
using System;
using System;

namespace Sitecore.Sharedsource.Web.HttpApplication
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Error(Object sender, EventArgs e)
        {
            Sitecore.Diagnostics.Log.Error(
                "Application exception in "
                + GetType().ToString() + " processing " + Sitecore.Context.RawUrl,
                System.Web.HttpContext.Current.Server.GetLastError(),
                this);
            Sitecore.Sharedsource.Web.UI.ErrorHelper helper =
                new Sitecore.Sharedsource.Web.UI.ErrorHelper();

            if (helper.ShouldRedirect())
            {
                helper.Redirect();
            }
        }
    }
}
```

6. Visual Studio のソリューション エクスプローラーで [すべてのファイルを表示] を有効にします。
7. Visual Studio のソリューション エクスプローラーで [`global.asax`] を右クリックし、[プロジェクトに含める] をクリックします。
8. Visual Studio のソリューション エクスプローラーで [すべてのファイルを表示] を無効にします。
9. 既存の `global.asax` ファイルのバックアップを取り、その内容を下記で置換し、名前空間とクラスの名前を適宜に更新します。

```
<%@ Application Language="C#"
    Inherits="Sitecore.Sharedsource.Web.HttpApplication.Global" %>
```

`Application_Error()` メソッドが例外のログを取り、「エラー ヘルパー」のセクションで記載されているように、さらにエラー処理を行うために、エラーヘルパーを使います。