Sitecore CMS 6.2

プレゼンテーション コンポーネント XSL リファレンス

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

この文書では Sitecore の XSL レンダリングで使用される XML、XSL、XPath の概念と構文の概要を解説します。[1] Sitecore の開発者、設計者には XSL レンダリング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 XSL の概要を説明し、XSL と XPath のコンストラクトの基礎を説明します。次に多くの XSL レンダリングが処理する Sitecore XML 構造体を説明します。続いて Sitecore のコンテキストにおける XSL と XPath について説明し、さらに XSL にカスタムの .NET 拡張機能を実装する方法について説明します。
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 — XSL と XPath のコンストラクトの基礎
- Chapter 3 — Sitecore の XML 構造体
- Chapter 4 — Sitecore での XSL と XPath
- Chapter 5 — カスタムの XSL 拡張ライブラリ

[1] XSL の詳細については、http://www.w3.org/Style/XSL/ を参照してください。XML の詳細については、http://www.w3.org/XML/ を参照してください。XPath の詳細については、http://www.w3.org/TR/xpath を参照してください。XSL レンダリングに関する詳細については、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20Reference.aspx から『プレゼンテーション コンポーネント リファレンス マニュアル』を参照してください。XSL レンダリングに関する情報は http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20Troubleshooting%20Guide.aspx から『プレゼンテーション コンポーネント トラブル シューティング ガイド』を参照してください。

Chapter 2

XSL と XPath のコンストラクトの基礎

この章では XML、XSL、XPath の概念と構文の基礎の概要を説明します。

この章ではまず XSL の概要を説明し、次に XML、XSL、XPath の特定のトークンについて説明します。次によく使われる XSL の要素を説明し、さらに XPath と XSL の機能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。

- XSL の概要
- XML、XSL、XPath のトークン
- XSL 要素
- XPath 関数と XSL 関数

2.1 XSL の概要

XSL は XML のデータ ソースを他のフォーマットに変換することを目的とした宣言型プログラミング言語です。変換先のフォーマットには HTML テキスト、他の XML、さらにはバイナリー フォーマットなどがあります。

XML はタグをもとにした、HTML のマークアップに似た階層構造のデータ ストアですが、より厳格な構文をもち、構造と内容では高い柔軟性を有しています。XSL は XML の一種の方言です。XSL ファイルは XSL 変換エンジンが認識できる要素を含む XML ファイルです。

XSL レンダリングは Sitecore データベースの XML 表現にアクセスします。Sitecore データベースの XML 表現に関する情報については、Chapter 3 「Sitecore の XML 構造体」を参照してください。

XSL のコンテキスト要素は、XML のドキュメントの XSL 変換エンジンの場所です。コンテキスト要素に関する詳細は、「コンテキスト要素」のセクションを参照してください。

XSL コードは XPath ステートメントを使って XML ドキュメントでノードを選択します。XPath に関する詳細は、「XML、XSL、XPath のトークン」と「XPath のナビゲーション」のセクションを参照してください。

XSL テンプレートはプロシージャーや関数のようなコードのブロックです。XSL テンプレートを名前を使って起動することができます。また特定の条件に合致する XML ドキュメントのノードを選択して、XSL テンプレートを起動することができます。XSL テンプレートに関する詳細は、「<xsl:template>、<xsl:call-template>、<xsl:with-param>、<xsl:apply-templates> XSL 要素」のセクションを参照してください。

XPath ステートメントは暗黙的と明示的な軸の使用を含みます。軸は XML ドキュメントをどのように読み取るかを指示します。軸に関する詳細は、「暗黙の関係（XPath 軸）」のセクションを参照してください。

重要

ASP.NET は XSL 1.0 をサポートします。ASP.NET と Sitecore は XSL 1.1 と XSL 2.0 はサポートしません。XSL 構文を調べる際には、XSL 1.1 または XSL 2.0 に固有な構造体の使用を避けてください。

重要

XSL のループは index 要素を `0` でなく `1` のポジションで始まるように構成します。

メモ

XSL をサポートしそれを使うために、ASP.NET を使用することができます。[2] ASP.NET は XSL で使用できる機能のスーパーセットを提供します。XSL または XPath の構文が扱いづらい場合、XSL のコードが管理しづらい場合、XSL のパフォーマンスが悪い場合、かなりのロジックを含むコンポーネントである場合、.NET を使用してください。

メモ

この文書では Sitecore コンテンツで機能する XSL 拡張機能に焦点を当てます。XML、XSL、XPath に関するさらなる情報は他の外部の文書を参照してください。[3]

[2] XSL と .NET レンダリング技術のメリットとデメリットに関する情報は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20Reference.aspx から『プレゼンテーション コンポーネント リファレンス マニュアル』、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』、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、それぞれ参照してください。

[3] Sitecore は次の文書をお勧めします： XSLT: Programmer's Reference、Michael Kay 著 (http://www.wrox.com/WileyCDA/WroxTitle/productCd-0764543814.html)

2.2 XML、XSL、XPath のトークン

この文書では、下記の表にあるような XSL レンダリングで通常よく使用されるトークンの詳細について解説します。

トークン	意味
.	１つのドット（点）文字は self 軸を表します。これは XML ドキュメントでの XSL 変換エンジンの現在の場所です。（コンテキスト要素とも呼ばれます）XML は self 軸、@id、./@id、self::@id、これらが皆、コンテキスト要素の id という名前の属性を選択するとみなします。同義語：self。
..	２つのドット（点）文字は parent 軸を表します。これはコンテキスト要素の親要素を含みます。同義語：parent.
/	スラッシュ文字はパス演算子です。XML ドキュメントのルート（/） を表し、要素をその属性 （./@id）と子要素（./item）から分離します。同義語：child.
//	２つのスラッシュ文字は descendant 軸を表します。これはコンテキスト要素のすべての子孫を選択します。同義語：descendant.
@	アットマーク文字は属性を指定します。たとえば、@key は key という名前の属性を選択します。
*	アスタリスク文字は任意の要素に合致します。たとえば、/* は任意の XML ドキュメントのルート要素を選択します。
$	ドル記号文字は名前付きのパラメーターまたは変数を示します。これは単なる値、または XML ドキュメントの場所です。たとえば、$sc_currentitem はコンテキスト アイテムを表します。
[]	角かっこ文字は XPath 述語を示します。これは要素の選択にフィルターをかけます。たとえば、./item はコンテキスト要素のすべての子を選択します。./item[@template='templatename'] は templatename という名前のテンプレートにもとづくコンテキスト要素の子を選択します。
::	２つのコロンは軸の解決演算子を表します。これは軸の名前を他のトークンから分離します。
&amp;	XML エスケープされたアンパサンド（"&"）。
&lt;	XML エスケープされた左山かっこ（"<"）。
&gt;	XML エスケープされた右山かっこ（">"）。

トークン	意味
`&quot;`	XML エスケープされた引用符（"）。
`&apos;`	XML エスケープされたアポストロフィー（〝'〞）。
`&####;`	16 進数で指定した XML エスケープされた文字。
`and`	and 論理演算子
`or`	or 論理演算子
`*`	**アスタリスク**文字は XML ドキュメントの任意のノードを選択できるワイルドカードです。

重要

パフォーマンスを最適化するため、`descendant` 軸の変形を使用するのは避けてください。

メモ

XML は上記のうち 5 つの名前付き文字エンティティのみをサポートします。XSL では、たとえば` ` などの HTML で使えるような、その他の名前付きエンティティを使うことはできません。代わりに、` `などのような、対応する数値エンティティを使います。[4]

[4] テキスト エンティティ コードと対応する数値のマッピングについては、http://www.webmonkey.com/reference/Special_Characters を参照してください。

2.3 XSL 要素

このセクションでは通常 Sitecore の XSL レンダリングで使用される XSL 要素の概要を説明します。XSL 要素は `xsl` 名前空間を使用します。

2.3.1 <xsl:variable> XSL 要素

`<xsl:variable>` XSL 要素は名前付き変数を作成します。たとえば、下記の例ではそれぞれ、変数 `$content` を作成します。これは `/Sitecore/Content` アイテム（Sitecore データベースを表す XML ドキュメントの `<item>` 要素）を表しています。

```
<xsl:variable name="content" select="/item[@key='sitecore']/item[@key='content']" />
<xsl:variable name="content" select="/*/item[@key='content']" />
<xsl:variable name="content" select="sc:item('/sitecore/content',.)"/>
```

最初のステートメントは完全修飾 XPath ステートメントを使って要素を選択しています。

２つめのステートメントではワイルドカードを使ってルート要素の子を選択しています。

３つめのステートメントでは .NET XSL 拡張メソッドを使って、指定したパスにある要素を選択しています。

Sitecore アイテムを表す変数を作成したら、そのアイテムから値を読み出し、さらにその子にも繰り返すか、または別の操作をアイテムに行うことができます。次の例では `$content` という名前の変数を作成し、その子に繰り返します。

```
<xsl:variable name="content" select="sc:item('/sitecore/content',.)"/>
<xsl:for-each select="$content/item">
  Child:<xsl:value-of select="@name" /><br />
</xsl:for-each>
```

できる限り、変数のオーバーヘッドを避けます。要素でなく ID を保存し、必要な場合には変数を使わずに直接 `<item>` 要素を参照します。たとえば、次の例では変数を使わずに前の例と同じ結果を出しています。

```
<xsl:for-each select="sc:item('/sitecore/content',.)/item">
  Child:<xsl:value-of select="@name" /><br />
</xsl:for-each>
```

メモ

XSL 変数の値を変えることはできません。変数をパラメーターとして再帰的に XSL テンプレートに渡すことでこの問題を回避できる場合があります。XSL テンプレートに関する詳細は、「<xsl:template>、<xsl:call-template>、<xsl:with-param>、<xsl:apply-templates> XSL 要素」のセクションを参照してください。

2.3.2 <xsl:value-of> XSL 要素

`<xsl:value-of>` XSL 要素は値を読み出します。`<xsl:value-of>` を使って値を出力ストリームに書き出すことができます。たとえば、**FieldName** という名前のフィールドの生の値を出力ストリームに書き出すには：

```
<xsl:value-of select="sc:fld('FieldName',$sc_currentitem)" />
```

`<xsl:value-of>` と名前付きテンプレートを使って変数に値を投入することができます：

```
<xsl:template name="GetContentID">
```

```
        <xsl:value-of select="sc:item('/sitecore/content',.)/@id" />
      </xsl:template>
      ...
      <xsl:variable name="contentid">
        <xsl:call-template name="GetContentID" />
      </xsl:variable>
```

disable-output-escaping 属性を使って、XML 特殊文字のエスケープを無効にします。たとえば、アンパサンド文字で区切られた複数のクエリ文字列パラメーターを含む URL を処理する場合に、出力のエスケープを無効にすることができます。

```
      <xsl:value-of
        select="sc:fld('FieldName', $sc_currentitem)" disable-output-escaping="yes" />
```

disable-output-escaping を yes に設定しない限り、XSL 変換エンジンは特殊文字をソースの値でエンコードします。たとえばシステムは、ソースの値でアンパサンド文字（"&"）には & を出力します。

2.3.3 <xsl:if> XSL 要素

<xsl:if> XSL 要素は、test 属性によって指定された条件が真（True）である場合に、含まれるコードを起動します。

XSL は Null と空の文字列を偽 （False）とみなすので、フィールドが存在し値を含んでいるかをチェックする簡単な方法は sc:fld() XSL 拡張メソッドが真を返すかどうかをチェックすることです。

```
      <xsl:if test="sc:fld('FieldName',$sc_currentitem)">
        <!-- FieldName exists and contains a value in the specified item-->
      </xsl:if>
```

メモ

XSL は <xsl:elseif> や <xsl:else> などの要素を含みません。複数の条件には、「<xsl:choose>、<xsl:when>、<xsl:otherwise> XSL 要素」のセクションで記載されているように、<xsl:choose>を使用します。

2.3.4 <xsl:choose>、<xsl:when>、<xsl:otherwise> XSL 要素

<xsl:choose> 要素は test 条件が真と評価される場合には最初の <xsl:when> 要素の内容を処理し、どの test も真と評価されない場合には <xsl:otherwise> 要素の内容を処理します。

```
      <xsl:variable name="random" select="sc:random(10)" />
      <xsl:choose>
        <xsl:when test="$random &gt; 6">
          <!--random is greater than 6-->
        </xsl:when>
        <xsl:when test="$random &gt; 3">
          <!--random is greater than three but less than or equal to 6-->
        </xsl:when>
        <xsl:otherwise>
          <!--random is less than or equal to 3-->
        </xsl:otherwise>
      </xsl:choose>
```

メモ

各 <xsl:choose> 要素に対し、システムは 1 つの <xsl:when> 要素または <xsl:otherwise> 要素だけに含まれているコードを処理します。<xsl:otherwise> 要素がなく、どの test 要素も真と評価されなかった場合は、システムは <xsl:choose> 要素のどのコードも処理しません。

2.3.5 <xsl:for-each> XSL 要素

<xsl:for-each> XSL 要素は、select 属性の XPath 表現に指定された 0 個または複数の要素に対して反復します。<xsl:for-each> 要素の各反復の中で、コンテキスト要素（".”）は選択された要素を含みます。

次の例ではコンテキストアイテムの子に対して反復し、それぞれの名前を出力します：

```
<xsl:for-each select="$sc currentitem/item">
  <xsl:value-of select="@name" /><br />
</xsl:for-each>
```

2.3.6 <xsl:sort> XSL 要素

<xsl:sort> XSL 要素は、<xsl:for-each>の中などの、要素のリストを並べ替えます。<xsl:for-each>に関する詳細は、「<xsl:for-each> XSL 要素」を参照してください。

次の例ではコンテキスト アイテムの子を日付フィールドの値で降順に並べ替えます。

```
<xsl:for-each select="$sc currentitem/item">
  <xsl:sort select="sc:fld('__updated',.)" order="descending" />
  <xsl:value-of select="@name" /><br />
</xsl:for-each>
```

@sortorder 属性を使ってデフォルトの並べ替え順を逆にすることができます。

```
<xsl:for-each select="$sc_currentitem/item">
  <xsl:sort select="@sortorder" order="descending" />
  <xsl:value-of select="@name" /><br />
</xsl:for-each>
```

メモ
デフォルトでは、アイテムは XML では、コンテンツツリーで見えるのと同じ順序で見えます。

2.3.7 <xsl:template>、<xsl:call-template>、<xsl:with-param>、<xsl:apply-templates> XSL 要素

<xsl:template> XSL 要素は XSL コードの一部をカプセル化します。これは他の言語でのメソッド、プロシージャー、関数に類似したものです。XSL 変換エンジンは <xsl:template> 要素の内容によって生成された出力を出力ストリームに書き出すか、またはその出力を XSL 変数に包みこむことができます。XSL ではテンプレートという用語は <xsl:template> 要素に含まれる XSL コードの一部を指します。

<xsl:call-template>を使ってテンプレートを起動すると、XSL 変換エンジンはテンプレートの出力を出力ストリームに書き出します。たとえば：

```
<xsl:template name="TemplateName">
  <xsl:value-of select="$sc currentitem/@template" />
</xsl:template>
<xsl:call-template name="TemplateName" />
```

または、テンプレートの出力を変数に配置することもできます。

```
<xsl:variable name="VariableName">
```

```
        <xsl:call-template name="TemplateName" />
      </xsl:variable>
```

<xsl:param> と <xsl:with-param> XSL テンプレートを使って、パラメーターを XSL テンプレートに渡すことができます。<xsl:param> の select 属性を使って、パラメーターのデフォルト値を指定することができます。

```
      <xsl:template name="TemplateName">
        <xsl:param name="ParamName" select="$sc currentitem" />
        <xsl:value-of select="$ParamName/@template" />
      </xsl:template>
      ...
      <xsl:call-template name="TemplateName">
        <xsl:with-param name="ParamName" select="."/>
      </xsl:call-template>
```

メモ

XSL テンプレートの中のコンテキスト要素は、呼び出しているコンテキストのコンテキスト要素です。

<xsl:apply-templates> XSL 要素を使い、アイテムの選択に XPath ステートメントを使って、XSL テンプレートを起動することもできます。

2.4 XPath 関数と XSL 関数

このセクションでは XSL プログラミングで使用される XPath ステートメントで利用できる関数の概要を説明します。

2.4.1 position() 関数

`position()` 関数はリストでの要素の場所を返します。`position()` に関する詳細は、「ancestor 軸と ancestor-or-self 軸」のセクションを参照してください。

よく使われる例は `position()` と `last()` を比較し、ループが最後の要素に達したかどうかを知ることです。たとえば、出力要素の間にスペース要素を挿入する場合などです。たとえば、次のコードでは HTML の改行（`<br>`）をコンテキスト アイテムのそれぞれの子アイテムへのリンクの後に出力します（ただしリストの最後のリンクを除く）。

```
<xsl:for-each select="$sc_currentitem/item">
  <sc:link><sc:text field="FieldName" /></sc:link>
  <xsl:if test="position() != last()">
    <br />
  </xsl:if>
</xsl:for-each>
```

2.4.2 last() 関数

`last()` 関数はリストでの要素の数を返します。`last()` 関数を使った例については、前のセクション「`position()` 関数」を参照してください。

2.4.3 current() 関数

`current()` 関数は XML ドキュメントの現在の要素を表します。`current()` 関数に関する詳細は、「現行の要素」を参照してください。

2.4.4 document() 関数

`document()` 関数は外部の XML ソースを読み出します。たとえば RSS フィードなどです。次のサンプル コードは RSS フィードを再整形します。

```
<xsl:variable name="rssurl" select="'http://www.asp.net/news/rss.ashx'" />
<xsl:variable name="rss" select="document($rssurl)" />
<xsl:if test="$rss">
  <xsl:for-each select="$rss//item">
    <a>
      <xsl:attribute name="href">
        <xsl:value-of select="link"/>
      </xsl:attribute>
      <xsl:value-of select="title"/>
    </a>
    <p>
      <xsl:value-of select="description" />
    </p>
```

```
      </xsl:for-each>
    </xsl:if>
```

メモ

RSS フィードの URL などの値をアイテムのフィールドに保存することができます。

```
    <xsl:variable name="rssurl" select="sc:fld('FieldName', $sc_currentitem)" />
```

2.4.5 concat() 関数

`concat()` 関数は文字列を連結します。

```
    <xsl:variable name="VariableName" select="concat('A', 'B')" />
    <xsl:variable name="VariableName" select="concat(concat('A', 'B'), 'C')" />
    <xsl:variable name="VariableName" select="concat(concat(concat('A', 'B'), 'C'), 'D')" />
```

2.4.6 translate() 関数

`translate()` 関数は文字を代替文字に変換するか、値から文字を削除します。たとえば、次のコードは **FieldName** という名前のフィールドの生の値を、ハイフン文字（〝-〞）をアンダースコア文字（〝_〞）で置換した後に、出力します。

```
    <xsl:value-of select="translate(sc:fld('FieldName', $sc_currentitem), '-', '_')" />
```

`translate()` 関数を使って、日付または日時の文字列表現を数値に変換することができます。Sitecore は日時の値を ISO の日付書式を使って保存しています。これは .NET の書式形式の `yyyyMMddTHHmmss` に対応します。`T` は値の日付部分を時刻部分と分離するリテラル文字です。比較またはその他の目的のために、この ISO 書式の日付を数値に変換するには、`translate()` 関数を使って `T` 文字を削除することができます。

```
    <xsl:variable name="updated"
      select="translate(sc:fld('__updated', $sc_currentitem), 'T', '')" />
```

2.4.7 true() 関数

`true()` 関数は真（True）の値を返します。この関数を使って値を逆にすることができます。

2.4.8 false() 関数

`false()` 関数は偽（False）の値を返します。この関数を使って値を逆にすることができます。

2.4.9 not() 関数

`not()` 関数は条件を否定します。次の条件は真となることがありません。

```
    <xsl:if test="not(true())">
      <!--the XSL transformation engine will never invoke this code.-->
    </xsl:if>
```

2.4.10 count() 関数

`count()` 関数はリストの要素の数を返します。たとえば、コンテキストアイテムが 5 つ以上の子の `<item>` 要素を持っているかどうかを知るには：

```
        <xsl:if test="count($sc_currentitem/item) &gt; 5">
```

2.4.11 contains() 関数

`contains()` 関数は、第 1 引数が第 2 引数を含んでいる場合に真を返します。

`contains()` 関数を使って ID のリスト（たとえば選択フィールドに保存されたリスト）が特定のアイテムの ID を含んでいるかどうかを知ることができます。たとえば、コンテキスト アイテムの **FieldName** フィールドがコンテキスト アイテムの ID を含んでいるかどうかを知るには：

```
        <xsl:variable name="ids" select="sc:fld('FieldName', $sc_currentitem/@id)" />
        <xsl:if test="contains($ids, $sc_currentitem/@id)">
          <!--The specified field in the context item contains the ID of the context item-->
        </xsl:if>
```

この条件を変数を作成することなく実装することができます。

```
        <xsl:if test="contains(sc:fld('FieldName', $sc_currentitem), $sc_currentitem/@id)">
          <!--The specified field in the context item contains the ID of the context item-->
        </xsl:if>
```

Chapter 3

Sitecore の XML 構造体

この章では XSL レンダリングが使用する Sitecore データベースの XML 表現の概要を解説します。

まず XML ドキュメントの中のアイテムへのアクセス方法を説明します。次に XPath を使って XML ドキュメントを探索し、ドキュメントの中のアイテムを選択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。

- アイテムを使う
- XPath のナビゲーション
- アイテムを選択

3.1 アイテムを使う

XSL レンダリングは Sitecore データベースの構造を表す XML ドキュメントにアクセスします。

重要
Sitecore データベースの生の XML 表現を調査するよりも、情報体系に関するお持ちの知識と、この文書に記載されている Sitecore の XSL 拡張機能を活用してください。

XML ドキュメントは `<item>` 要素の階層構造から構成されています。各 `<item>` 要素は Sitecore のコンテンツ ツリーのアイテムに対応します。各 `<item>` 要素は属性を持ち、アイテムのフィールドを定義する要素のコレクションを含んでいます。

各 `<item>` は子の `<item>` 要素をいくつでも含むことができます。XML ドキュメントは、コンテンツ ツリーでアイテムをネストしたものと同じ深さまでネストされた `<item>` 要素を含みます。

XML ドキュメントのルート `<item>` 要素は、コンテンツ ツリーの `/Sitecore` アイテムを表します。次の XML の断片は、デフォルトの `/Sitecore` と `/Sitecore/Content` アイテムのデータの一部です。`<item>` 要素はすべて類似した構造になっています。

```
<item name="sitecore" key="sitecore" id="{11111111-1111-1111-1111-111111111111}"
  tid="{C6576836-910C-4A3D-BA03-C277DBD3B827}"
  mid="{00000000-0000-0000-0000-000000000000}"
  sortorder="100" language="en" version="1" template="root"
  parentid="{00000000-0000-0000-0000-000000000000}">
  <fields>
    <field tfid="{5DD74568-4D4B-44C1-B513-0AF5F4CDA34F}" key="__created by"
    type="text" />
    <!-- additional field definition elements -->
   <field tfid="{9C6106EA-7A5A-48E2-8CAD-F0F693B1E2D4}" key="__read only"
    type="checkbox" />
  </fields>
  <item name="content" key="content" id="{0DE95AE4-41AB-4D01-9EB0-67441B7C2450}"
  tid="{E3E2D58C-DF95-4230-ADC9-279924CECE84}"
  mid="{00000000-0000-0000-0000-000000000000}"
  sortorder="100" language="en" version="1" template="main section"
  parentid="{11111111-1111-1111-1111-111111111111}">
    <fields>
      <field tfid="{BADD9CF9-53E0-4D0C-BCC0-2D784C282F6A}" key="__updated by"
      type="text"  />
      <!-- additional field definition elements -->
      </field>
    </fields>
    <!— item elements at this level represent children of /Sitecore/Content-->
    ...
  </item>
  <!-- additional item elements at this level represent siblings of /Sitecore/Content-->
</item>
```

3.1.1 アイテムの属性

各 <item> 要素は特定の属性のセットを持っています。それには下記のようなものがあります。

- **`@name`**: アイテムの名前。

- **@key**: 小文字のアイテムの名前。
- **@id**: アイテムの ID。
- **@tid**: アイテムに関連付けられたデータ テンプレート定義アイテムの ID。
- **@mid**: アイテムを挿入するために使用されたブランチのテンプレートまたはコマンド テンプレートの ID、または null GUID。
- **@sortorder**: シブリング（兄弟）アイテムに相対的なアイテムの数値による並べ替え順。
- **@language**: コンテキスト言語。
- **@version**: その言語内でのアイテムのバージョン数。
- **@template**: アイテムに関連付けられたデータ テンプレートの小文字の名前。
- **@parentid**: アイテムの親の ID、またはルートアイテムの null GUID。

注意
良いパフォーマンスのために、常に `@id` 属性の値を比較し、2 つの要素が同じアイテムを表していないかどうか、調べてください。たとえば、`$sc_currentitem = $sc_item` でなく、`$sc_currentitem/@id = $sc_item/@id` をテストしてください。

3.1.2 アイテムのフィールド

Sitecore データベースの XML 表現の各 `<item>` 要素は 1 つの `<fields>` 要素を含みます。各 `<fields>` 要素はいくつかの `<field>` 要素を含みます。各 `<field>` 要素はアイテムに関連付けられたデータ テンプレートのフィールド定義、またはそのテンプレートに関連付けられたベース テンプレートの 1 つ（標準テンプレートを含む）を表します。

メモ
効率のため、Sitecore データベースの XML 表現はフィールド値を含んでいません。`<field>` 要素は値を含みません。フィールド値にアクセスするために使用できる XSL 拡張機能に関する詳細は、次のセクションを参照してください：「`<sc:date>` XSL 拡張コントロール」、「`<sc:html>` XSL 拡張コントロール」、「`<sc:image>` XSL 拡張コントロール」、「`<sc:link>` XSL 拡張コントロール」、「`<sc:memo>` XSL 拡張コントロール」、「`<sc:text>` XSL 拡張コントロール」、「`<sc:wordStyle>` XSL 拡張コントロール」、「`sc:fld()` XSL 拡張メソッド」、「`sc:field ()` XSL 拡張メソッド」。

各 `<field>` 要素は特定の属性のセットを持っています。それには下記のようなものがあります。

- **@tfid**: データ テンプレート フィールド定義アイテムの ID。
- **@key**: データ テンプレート フィールド定義アイテムの小文字の名前。
- **@type**: データ テンプレート フィールド データ タイプの小文字の名前。

3.2 XPath のナビゲーション

XPath を使って XSL レンダリングで使用できる XML ドキュメントを探すことができます。

3.2.1 特定のアイテム

XSL レンダリングはアイテムにアクセスするために使用できるパラメーターを定義します。

コンテキスト アイテム： $sc_currentitem

コンテキスト アイテムはクライアントに要求された URL のパスに対応するアイテムです。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すべてのレンダリングのデフォルトのデータ ソースです。`$sc_currentitem` 変数は、Sitecore データベースの XML 表現のコンテキスト アイテムに対応する `<item>` 要素を表します。

データ ソース アイテム： $sc_item

レンダリングはデータ ソース アイテムからデータを読み出すことができます。`$sc_item` 変数は XSL レンダリングに対してデータ ソース アイテムを表します。開発者がレンダリングに対してデータ ソースを指定しない場合、デフォルトのデータ ソース アイテムはコンテキスト アイテムで、`$sc_item` と `$sc_currentitem` は同じアイテムになります。

XSL レンダリングのロジックはコード ファイルの最初の `<xsl:template>` XSL 要素で始まります。XSL レンダリングの定型コード ファイルは `<xsl:apply-templates>` XSL 要素の `select` 属性を使い、コンテキスト要素をデータ ソース アイテムに設定し、`mode` 属性 `main` で XSL テンプレートを起動します。

```
<xsl:template match="*">
  <xsl:apply-templates select="$sc_item" mode="main"/>
</xsl:template>
<xsl:template match="*" mode="main">
  <!--the context element is the data source item-->
  <!--developers typically insert code here-->
</xsl:template>
```

コンテキスト要素

コンテキスト要素は、XML のドキュメントの中の XSL 変換エンジンの場所です。コンテキスト要素は、XSL 変換エンジンがそこから相対 XPath ステートメントを解釈する場所です。ドット文字（“.”）がコンテキスト要素を表します。`<xsl:for-each>`などの XSL コンストラクトはコンテキスト要素を変更しますが、コンテキスト アイテム（`$sc_currentitem`）は変更しません。

現行の要素

`current()` 関数は現行の要素を返します。これを XPath 述語を使って、現行のスコープのコンテキスト要素にアクセスすることができます。XPath ステートメントの述語の中で、現行の要素は、XSL 変換エンジンが XPath ステートメントの評価を

始めた時点でコンテキスト要素であった要素です。その他の点では、現行の要素とコンテキスト要素は多くの場合同じ要素です。次のサンプルコードを考察します：

```
<xsl:for-each select="$sc_item/item">
  <xsl:choose>
    <xsl:when test="$sc_currentitem/ancestor-or-self::item[@id=current()/@id">
      <!—the context element is the iteration item or one of its descendants-->
    </xsl:when>
  </xsl:choose>
</xsl:for-each>
```

外側の `<xsl:for-each>` 要素が データ ソース アイテムの子である `<item>` 要素に反復されています。外側の `<xsl:for-each>` の中で、コンテキスト要素はデータ ソース アイテムの子で、`current()` 関数はそのアイテムを返します。`<xsl:when>` 要素は、コンテキスト アイテム（`$sc_currentitem`）がコンテキスト要素（`current()`）としての祖先であるか、またはそれをもつかをテストします。たとえば、コンテキスト アイテムが情報体系のそのセクションの中にあるかどうかを見きわめます。述語の中で、コンテキスト要素は `ancestor-or-self` 軸上の `<item>` 要素であり、一方、現行の要素は引き続きデータ ソース アイテムの子です。

重要

コンテキスト アイテム、コンテキスト要素、現行の要素の違いを理解することは重要です。これらは皆同じ `<item>` を表すこともあります。コンテキスト アイテムとはブラウザーによって要求されたアイテムです。コンテキスト要素は、XML のドキュメントの中の XSL 変換エンジンの場所です。コンテキスト アイテムは明示的なデータ ソースを持たないレンダリングのためのデフォルトのコンテキスト要素です。コンテキスト要素は通常は Sitecore データベースを表す XML ドキュメントの中の `<item>` ですが、任意の型の要素である場合もあり、別の XML ドキュメントにある場合もあります。現行の要素はコンテキスト要素ですが、ループしたコンストラクトの述語にある場合を除きます。その場合は現行の要素はループが始まる時にコンテキスト要素であった要素です。

XPath を使ったアイテム変数

完全修飾 XPath ステートメントを使って任意のアイテムを参照することができます。たとえば、次の XSL コンストラクトを使って `/Sitecore/Content` アイテムを表す変数を作成することができます：

```
<xsl:variable name="content" select="/item[@key='sitecore']/item[@key='content']" />
```

メモ

XML ドキュメントは常に必ず 1 つのルート要素を持つので、この表現をアスタリスク文字（〝*〟、ワイルドカード）を使ってルート要素に合致させるようにして、短縮することができます。

```
<xsl:variable name="content" select="/*/item[@key='content']" />
```

sc:item() を使ったアイテム変数

`sc:item()` XSL 拡張メソッドにパスまたは ID を渡して、アイテムにアクセスすることができます。

```
<xsl:variable name="content" select="sc:item('/sitecore/content',.)"/>
<xsl:variable name="content"
  select="sc:item('{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}',.)"/>
<xsl:for-each select="$content/item">
  <xsl:value-of select="@name" /><br />
</xsl:for-each>
```

<xsl:param> XSL 要素を使ってアイテムを XSL レンダリングに渡す

<xsl:param> XSL 要素を使って、さらなるアイテムのパスまたは ID を XSL レンダリングに渡すことができます。sc:item() XSL 拡張メソッドを使って対応するアイテムを選択します。[5] レンダリング パラメーター定義を XSL レンダリングのヘッダーのデフォルトのパラメーター定義の付近に追加します。

```
<xsl:param name="ParamNamePathOrID"><!--default parameter value--></xsl:param>
<xsl:variable name="VariableName" selet="sc:item($ParamNamePathOrID, .)"/>
```

3.2.2 アイテムの参照

1 つのアイテムは他の Sitecore アイテムの ID を含むフィールドを含み、1 つのアイテムから別のアイテムへの参照を表すことができます。

1 つのフィールドが 1 つの ID を含む場合、sc:fld() XSL 拡張メソッドを使ってその ID を読み出し、その ID を sc:item() XSL 拡張メソッドに渡して、対応する <item> 要素を選択することができます。

```
<xsl:variable name='IDVariableName' select='sc:fld('FieldName", $sc_item)' />
<xsl:if test="$IDVariableName">
  <xsl:variable name='ItemVariableName' select='sc:item($IDVariableName, $sc_item)' />
  <xsl:if test="$ItemVariableName">
    <xsl:value-of select="$ItemVariableName/@name" />
  </xsl:if>
</xsl:if>
```

1 つのフィールドが複数のアイテムを参照している ID のリストを含む場合、sc:Split() XSL 拡張メソッドを使ってそのリストに対して反復することができます。

```
<xsl:for-each select="sc:Split('FieldName',.)">
  <xsl:for-each select="sc:item(text(),.)">
    <xsl:value-of select="@name" /><br />
  </xs:for-each>
</xsl:for-each>
```

メモ

<xsl:for-each>を使って sc:item() XSL 拡張関数によって返されたアイテムを処理することで、コンテキスト要素を設定し、短く、柔軟で、一貫性のある構文になります。

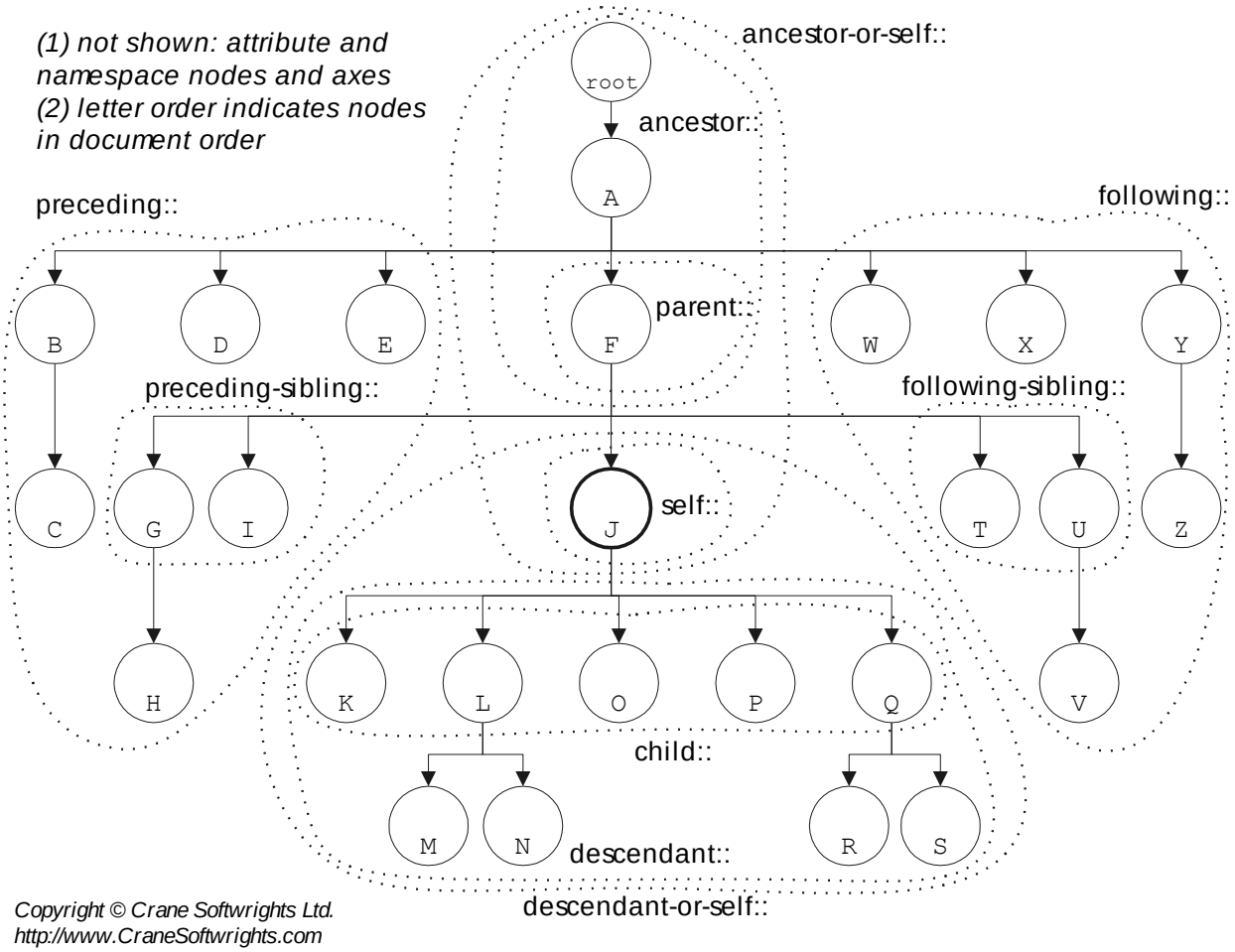
ヒント

sc:Split() XSL 拡張メソッドを使って 1 つのアイテムの ID を含むフィールドを処理することができ、また複数のアイテムの ID を含むフィールドも処理することができます。

[5] パラメーターをレンダリングに渡す方法の詳細については、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Presentation%20Component%20Reference.aspx から『プレゼンテーション コンポーネント リファレンス マニュアル』を参照してください。

3.2.3 暗黙の関係（XPath 軸）

Sitecore データベースを表す XML ドキュメントの各 <item> 要素は、各種の XPath 軸を通して、そのドキュメントの他の <item> 要素といくつかの暗黙の関係を持っています。次のダイアグラムでは J がコンテキスト要素を表し、各種の XPath 軸の概要を示しています。[6]



self 軸

self 軸はコンテキスト要素のみを含みます。XPath 表現ではドット文字（".”）は self 軸を表します。self 軸は明示的に軸を指定しない場合のデフォルトの軸です。開発者は通常は self 軸を明示的に使用するより、暗黙的または明示的にコンテキスト要素を参照します。次の 3 つのコンストラクトは等価です。一般には短いほど好ましいでしょう。

```
<xsl:value-of select="@id" />
<xsl:value-of select="./@id" />
<xsl:value-of select="self::*/@id" />
```

[6] Crane Softwrights Ltd., http://www.CraneSofwrights.com の許可により掲載。

最初の例の select 属性の XPath ステートメントは軸を指定していないので、暗黙的にコンテキスト要素の属性の値を読み出します。2 つめの例では明示的にコンテキスト要素（.）を参照し、パス演算子（/）を使ってそのアイテムの名前付き属性を選択しています。3 つめの例は self 軸（self::）上の任意の要素（*）に明示的に合致し、そのような任意の要素の id という名前の属性の値を読み出します。

child 軸

child 軸はコンテキスト要素の子を含みます。XPath 表現ではパス演算子（〝/〟）は child 軸を表します。

各 <item> 要素は、アイテムの子を表すネストされた <item> 要素を含むことができます。アイテムの子を選択するときに <fields> 要素を除外するには、明示的に子要素の名前の付いたアイテムを選択します。次のそれぞれの XPath ステートメントでは item トークンが <item> 要素に合致します。

```
<xsl:for-each select="./item">
<xsl:for-each select="item">
<xsl:for-each select="./child::item" />
<xsl:for-each select="child::item">
```

最初の例ではコンテキスト要素の子である <item> 要素に合致します。2 つめの例では同じアイテムに合致し、XPath がデフォルトでコンテキスト要素から child 軸を処理することを示しています。3 つめの例ではコンテキスト要素の child 軸上の <item> 要素に合致します。4 つめの例ではコンテキスト要素の child 軸上の <item> 要素に合致します。

次のようなコンストラクトを使ってアイテムに子があるかどうか知ることができます：

```
<xsl:if test="$sc currentitem/item">
  <!--the context item has child items-->
</xsl:if>
```

parent 軸

parent 軸はコンテキスト要素の親を含みます。ルート要素を除き、XML ドキュメントの各要素は 1 つの親要素を持ちます。XPath 表現では 2 つのドット文字（〝..〟）は parent 軸を表します。

次の例では、コンテキスト要素の親アイテムに関連付けられたデータ テンプレートの名前が homepage であるかどうかをテストします。

```
<xsl:if test="./parent::item[@template='homepage']">
<xsl:if test="parent::item[@template='homepage']">
<xsl:if test="parent::*[@template='homepage']">
<xsl:if test="./../[@template='homepage']">
<xsl:if test="../[@template='homepage']">
```

メモ

次の表現では、親アイテムが指定されたデータ テンプレートを使うかどうかではなく、親アイテムがそのデータ テンプレートを使う子を持つかどうかを知ることができます。

```
<xsl:if test="../item[@template='homepage']">
```

ancestor 軸と ancestor-or-self 軸

ancestor 軸はコンテキスト要素の祖先をドキュメント順に返します。ドキュメント順とは、XML ドキュメントでのルート要素から始まる要素の順序のことです。ancestor-or-self 軸はコンテキスト要素とその祖先をドキュメント順に返します。

「暗黙の関係（XPath 軸）」のセクションで示したダイアグラムで、コンテキスト アイテムが S である場合、ancestor 軸は A、F、J、Q を含みます。また ancestor-or-self 軸は A、F、J、Q、S を含みます。

XSL レンダリングは多くの場合に ancestor 軸と ancestor-or-self 軸を使って、他のアイテムを含むアイテムを処理したり、あるアイテムが他のアイテムの子孫であるかどうかを把握して、ブレッドクラムを生成したり、コンテキスト アイテムを含むアイテムに対応するナビゲーション要素をハイライトします。

<xsl:for-each> 要素の select 属性の中で、position() 関数は要素のリストの中の要素のインデックスを参照します。ancestor 軸と ancestor-or-self 軸では、これは XML ドキュメント順の逆です。<xsl:for-each> 要素の select 属性の述語では、last() 関数は処理する要素のリストの中の要素の数を返します。

ancestor 軸は多くの場合ブレッドクラムで役立ちます。ブレッドクラムは A と F へのリンクを含むべきではありません。これはそれぞれ /Sitecore と /Sitecore/Content アイテムを表します。ブレッドクラムは通常はコンテキスト アイテムの名前を含みますが、ブレッドクラムのそのステップは通常はリンクではありません。ancestor 軸はコンテキスト アイテムを除外します。A と F を除くには、コンテキスト アイテムから最も遠い 2 つの要素を除外します。

```
<xsl:for-each select="$sc_currentitem/ancestor::item[position() &lt; last() - 1]">
```

ancestor-or-self 軸を使って、あるアイテムが他のアイテムの祖先またはそれ自身であるかどうかを知ることができます。たとえば、あるレンダリングがホーム アイテムの下にあるセクションに反復し、各セクションへのリンクを生成し、現行のページを含むセクションをハイライトすることができます。

```
<xsl:for-each select="$home/item[@template='section']">
  <xsl:choose>
    <xsl:when test="$sc_currentitem/@id=@id">
      <!--the client has requested this section item-->
    </xsl:when>
    <xsl:when test="$sc currentitem/ancestor-or-self::item[@id=current()/@id">
      <!--the client has requested an item that is a descendant of this item-->
    </xsl:when>
    <xsl:otherwise>
      <!--the client has not requested this item or any of its descendants-->
    </xsl:otherwise>
  </xsl:choose>
</xsl:for-each>
```

ancestor-or-self 軸を使って、あるアイテムまたはそのフィールドに値を含む最も近い祖先から、フィールド値を読み出すことができます。次のコードは、コンテキスト アイテムの **FieldName** という名前のフィールド、またはそのフィールドに値を含む最も近い祖先を処理します。

```
<sc:image field="FieldName"
  select="$sc_currentitem/ancestor-or-self::item[sc:fld('FieldName',,'src')][1]" />
```

トークン $sc_currentitem/ancestor-or-self::item により XSL 変換エンジンが、コンテキスト要素とその祖先の <item> 要素のそれぞれに対して、述語を評価します。最初の述語が選択を、**FieldName** フィールド（[sc:fld('FieldName',,'src')]）の値を定義する <item> 要素のみに制限します。2 つめの述語（[1]）

が、合致してコンテキスト アイテムに最も近い <item> 要素を選択します。<sc:image> コントロールが、そのアイテムで指定されたフィールド値を使って HTML <img> タグを生成します。

子孫と再帰

descendant 軸は要素のすべて子孫を、再帰的に、ドキュメント順に、含みます。descendant-or-self 軸はコンテキスト要素とそのすべての子孫を、再帰的に、ドキュメント順に、含みます。

「暗黙の関係（XPath 軸）」のセクションで示したダイアグラムで、コンテキスト要素が J である場合、descendant 軸は K、L、M、N、O、P、Q、R、S の各要素を含みます。また descendant-or-self 軸は J、K、L、M、N、O、P、Q、R、S の各要素を含みます。

descendant 軸と descendant-or-self 軸で使われるものをはじめ、再帰は高コストですが、特に少量のデータの場合には効果的です。次の例ではデータ駆動型のサイトマップを生成します。

```
        <xsl:call-template name="SiteMapStep" />
        ...
        <xsl:template name="SiteMapStep">
          <xsl:param name="level" select="1" />
          <xsl:param name="start" select="$home" />
          <ul class="{concat('sitemap',$level)}">
            <xsl:for-each select="$start/item">
              <li>
                <sc:link><sc:text field="title" /></sc:link>
                <xsl:if test="item">
                  <xsl:call-template name="SiteMapStep">
                    <xsl:with-param name="level" select="$level+1" />
                    <xsl:with-param name="start" select="."/>
                  </xsl:call-template>
                </xsl:if>
              </li>
            </xsl:for-each>
          </ul>
        </xsl:template>
```

注意

パフォーマンスの考慮のため、descendant 軸と descendant-or-self 軸の過度な使用は避けてください。// コンストラクトも同様です。特に多数のアイテムを処理する場合には注意します。単一行のコードにより処理されるアイテムの数を制限するように、情報体系とユーザーの検索インデックスとその他の機能を構築してください。

3.3 アイテムを選択する

このセクションでは処理するアイテムを選択するテクニックを説明します。

3.3.1 特定のデータ テンプレートにもとづきアイテムを選択する方法

XPath の述語の `@template` 属性を使って、特定のデータ テンプレートにもとづくアイテムを処理することができます。

```
<xsl:for-each select="$sc_item/item[@template='templatename']">
```

重要

`@template` 属性はデータ テンプレート キーを含みます。これはデータ テンプレートの小文字の名前です。

1 つまたは複数のテンプレートにもとづくアイテムを処理するには、XPath の `contains()` 関数の使用を検討します。

```
<xsl:for-each select="$sc_item/item[contains('!templatename1!templatename2!',
  concat(concat('!',@template),'!')]">
```

共通のベース テンプレートを共用するデータ テンプレートにもとづくアイテムを処理する方法:

```
<xsl:for-each select="$sc_currentitem/item[sc:IsItemOfType('basetemplate',.)]">
```

直接ベース テンプレートをもとにしているアイテムを含め、共通のベース テンプレートを共用するデータ テンプレートにもとづくアイテムを処理するには：

```
<xsl:for-each select="$sc_currentitem/item[@template='basetemplate' or
  sc:IsItemOfType('basetemplate',.)]">
```

3.3.2 コンテキスト言語でバージョンつきのアイテムを選択する方法

複数言語をサポートするサイトでは、CMS ユーザーはパブリッシュの前にすべてのアイテムを翻訳するわけではありません。コンテキスト言語のバージョンを持つアイテムを選択するには、作成日付フィールドで値をクリックします。たとえば：

```
<xsl:for-each select="$sc_currentitem/item[sc:fld('__created',.)]">
  <!--the context element is an item with a version in the context language-->
</xsl:for-each>
```

3.3.3 子を持つアイテムを選択する方法

子を持つアイテムを選択するには、子 `<item>` 要素の存在を指定する述語を含めます。たとえば、少なくとも 1 つの子アイテムを持つ、コンテキスト アイテムのすべての子を処理するには：

```
<xsl:for-each select="$sc_currentitem/item[item]">
  <!--the context element, a child of the context item, has at least one child item-->
</xsl:for-each>
```

コンテキスト言語のバージョンを持つ特定のデータテンプレートをもとにした、または継承した、少なくとも 1 つの子アイテムを持つ、コンテキスト アイテムのすべての子を処理する方法:

```
<xsl:for-each select="$sc currentitem/item[item[@template='basetemplate' or
  sc:IsItemOfType('basetemplate',.)) and sc:fld('__created',.)]]">
```

Chapter 4

Sitecore での XSL と XPath

この章では XSL、XPath、.NET で書かれた Sitecore XSL 拡張機能など、XSL レンダリングを取り扱う開発者のための考慮点とテクニックについて説明します。

まず、新規に XSL レンダリングを作成するために使用できる定型コード ファイルについて説明します。次に Sitecore が XSL レンダリングのエラーをどのように処理するかを説明します。続いて Sitecore のアイテムのフィールドにアクセスする方法を説明します。さらに XSL 拡張メソッドと XSL 拡張コントロール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 XSL 定型コード ファイル
- XSL エラー処理
- フィールドを取り扱う
- XSL 拡張コントロールとメソッドの概要
- Sitecore の XSL 拡張コントロール
- Sitecore の XSL 拡張コントロール

4.1 Sitecore の XSL 定型コード ファイル

デベロッパー センターで新規の XSL レンダリングを作成する場合、Sitecore は XSL レンダリングの定型コード ファイルを複製します。これは新しいコードのための出発点になります。

新規の XSL レンダリングのための定型コードファイルは下記の内容を含みます。

```
<?xml version="1.0" encoding="UTF-8"?>
```

この行はファイルが XML を含んでいることを示しています。XSL レンダリングファイルは XSL コードを含みます。XSL は XML の一種の方言で、すべての XSL ファイルは XML ファイルです。

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:sc="http://www.sitecore.net/sc"
  xmlns:dot="http://www.sitecore.net/dot"
  exclude-result-prefixes="dot sc">
```

この行が XSL ファイルの XML ルート要素を含み、XSL コードとしてのファイルの内容を特定します。ASP.NET がタグ プリフィックスをコントロールを含むアセンブリにマッピングするように、XSL は名前空間の識別子を XSL 処理機能を含む URL にマッピングします。`xsl` 名前空間が XSL 言語を公開します。Sitecore はデフォルトでは `sc` と `dot` 名前空間を定義します。これは Sitecore によってインストールされたアセンブリの .NET クラスに対応します。`web.config` の `/configuration/sitecore/xslExtension/extension` 要素は各クラスの署名を提供し、これによって XSL 名前空間を .NET アセンブリにマッピングします。.NET XSL 拡張機能に関する詳細は、Chapter 5 「カスタムの XSL 拡張ライブラリ」を参照してください。

```
<xsl:output method="html" indent="no" encoding="UTF-8" />
```

この行はレンダリングが HTML 構文を使ってマークアップを出力することを示しています。これは <hr> とその他の要素の閉じる要素を必要としません。XHTML サイトには、`method` 属性の値は `xml` であるべきで、`<hr />` または `<hr></hr>` などの出力となるべきです。

```
<xsl:param name="lang" select="'en'"/>
<xsl:param name="id" select="''"/>
<xsl:param name="sc item"/>
<xsl:param name="sc_currentitem"/>
```

これらの行は Sitecore がすべての XSL レンダリングに渡すいくつかのパラメーターを定義します。

- **`$lang`**: コンテキスト言語。
- **`$id`**: レンダリングのデータ ソース アイテムの GUID。
- **`$sc_item`**: レンダリングのデータ ソース アイテム。
- **`$sc_currentitem`**: コンテキスト アイテム。

```
<!--<xsl:variable name="home" select="sc:item('/sitecore/content/home',.)"/>-->
<!--<xsl:variable name="home" select="/*/item[@key='content']/item[@key='home']" />-->
<!--<xsl:variable name="home"
  select="$sc_currentitem/ancestor-or-self::item[@template='site root']" />-->
```

これらの行はコンテキスト サイトのホーム アイテムにアクセスする 3 つの方法を示しています。[7]

```
        <xsl:template match="*">
          <xsl:apply-templates select="$sc_item" mode="main"/>
        </xsl:template>
```

これらの行は、mode 属性に main の値を使って次の XSL テンプレートを起動する前に、コンテキスト要素をレンダリング（$sc_item）のデータ ソースに設定します。

```
        <xsl:template match="*" mode="main">

        </xsl:template>
```

開発者は通常はこの <xsl:template> 要素の中にコードを追加します。

```
        </xsl:stylesheet>
```

この行は <xsl:stylesheet> ルート要素を閉じます。

[7] ロジックを使ってコンテキスト サイトのホーム アイテムを知る方法の例については、http://trac.sitecore.net/XslHelper/browser/Trunk/Xml/Xsl/XslHelper.cs から GetHomeItem() メソッドを参照してください。

4.2 XSL エラー処理

レイアウト エンジンで構文エラーが発生した場合または XSL レンダリングを処理している間に例外が発生した場合、レイアウト エンジンはエラーに関する情報を出力ストリームに埋め込み、それは Web クライアントに表示されます。[8]

レイアウト エンジンは XSL レンダリングを実行時までコンパイルしません。XSL 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。

.NET で書かれた XSL の拡張機能は例外をスローすることができます。XSL レンダリングが例外をスローする拡張メソッドを呼び出した場合、レイアウト エンジンはその例外に関する情報を出力ストリームに追加し、それは Web クライアントに表示されます。この出力はレンダリングによって例外の発生以前に生成されたすべての出力の後に表示されます。XSL レンダリングは例外の発生以後は出力ストリームに書き込みません。

[8] Sitecore のエラー処理に関する詳細、およびエラー処理の上書き方法については、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。

4.3 フィールドを取り扱う

次の表は XSL レンダリングの各種のデータ テンプレート フィールド タイプにアクセスするために使用できる Sitecore の XSL 拡張コントロールとメソッドをまとめています。さらにこの文書の後のセクションで詳細に説明します。

フィールド タイプ	コンストラクト	メモ
Attachment Icon IFrame Integer Internal Link Layout Number Password Security Template Field Source Tristate	なし	これらのフィールド タイプを処理するために XSL を使わないでください
Checkbox Droplist File Grouped Droplist	`sc:fld()`	`sc:fld()` XSL 拡張メソッドを使ってこれらのフィールド タイプを処理することができます。
Checklist Multilist Treelist TreelistEx	`sc:Split()` と `sc:item()`	`sc:Split()` XSL 拡張メソッドを `sc:item()` XSL 拡張メソッドと使用して、ユーザーが複数のアイテムを選択できるフィールドで参照されているアイテムを読み出すことができます。
Date DateTime	`<sc:date>`	`<sc:date>` XSL 拡張コントロールを使って日付と日時フィールドを処理することができます。
Droplink Droptree Grouped Droplink Internal Link	`sc:fld()` と `sc:item()`	`sc:item()` XSL 拡張メソッドを `sc:fld()` XSL 拡張メソッドと使用して、ユーザーが単一のアイテムを選択できるフィールドで参照されているアイテムを読み出すことができます。

フィールド タイプ	コンストラクト	メモ
File Drop Area (FDA)	`sc:fld()` と `sc:item()`	`sc:fld()` XSL 拡張メソッドを使って FDA フィールドの mediaid 属性を読み出すことができます。`sc:item()` XSL 拡張メソッドを使ってそのアイテムの子にアクセスすることができます。FDA フィールドへのアクセスに関する詳細は、次の「FDA （ファイル ドロップ エリア） フィールド」のセクションを参照してください。
General Link	`<sc:link>`	`<sc:link>` XSL 拡張コントロールを使って一般のリンク フィールドを処理することができます。
Image	`<sc:image>`	`<sc:image>` XSL 拡張コントロールを使ってイメージ フィールドを処理することができます。
Multi-Line Text	`<sc:memo>`	`<sc:memo>` XSL 拡張コントロールを使って複数行テキストフィールドを処理することができます。
Rich Text Editor (RTE)	`<sc:html>`	`<sc:html>` XSL 拡張コントロールを使ってリッチ テキスト エディター フィールドを処理することができます。
Single-Line Text	`<sc:text>`	`<sc:text>` XSL 拡張コントロールを使って単一行テキストフィールドを処理することができます。
Word Document	`<sc:text>`	`<sc:text>` XSL 拡張コントロールを使ってワード文書フィールドを処理することができます。

メモ

`sc:fld()` と `sc:field()` XSL 拡張メソッドを使って任意のタイプのフィールドにアクセスすることができます。

メモ

`<sc:wordstyle>` XSL 拡張要素を使って、Word Document 型の各フィールドをサポートするために必要な CSS スタイルを含む HTML <style> 要素を埋め込むことができます。`<sc:wordStyle>` XSL 拡張コントロールの詳細については、「`<sc:wordStyle>` XSL 拡張コントロール」のセクションを参照してください。

ヒント
Image、File、General Link フィールドで使用できる属性など、フィールドに保存された値を参照するには、生のフィールド値を参照します。[9]

4.3.1 FDA（ファイル ドロップ エリア） フィールド

次のサンプル コードをもとにしてコードを実装することができます。この例では **FileDropAreaField** という名前の FDA（ファイルド ロップ エリア）フィールドのメディア アイテムにアクセスします。

```
        <xsl:variable name="folderid"
          select="sc:fld('filedropareafield', $sc_item, 'mediaid')" />
        <xsl:if test="$folderid and sc:item($folderid, $sc item)/item">
          <xsl:for-each select="sc:item($folderid, $sc_item)/item">
            <a href="{sc:GetMediaUrl(.)}">
              <xsl:value-of select="@name"/>
            </a>
          </xsl:for-each>
        </xsl:if>
```

`$folderid` 変数は、レンダリングのデータ ソースの **FileDropAreaField** という名前の FDA フィールドの `mediaid` という名前の属性の GUID を含みます。**FileDropAreaField** の値が空でなく、対応するアイテムに子がある場合、このコードはそれらの子のそれぞれへのリンクを生成します。

[9] 生のフィールド値にアクセスする方法についての詳細は、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。

4.4 XSL 拡張コントロールとメソッドの概要

XSL 拡張機能は .NET ロジックを XSL レンダリングに公開します。XSL 拡張機能には 2 つの種類があります。XSL 拡張コントロールと XSL 拡張メソッドです。

XSL 拡張コントロールは .NET クラスに対応する XSL レンダリングの中の XML 要素です。たとえば、`<sc:text>` XSL 拡張コントロールは `Sitecore.Web.UI.XslControls.Text` .NET クラスに対応します。XSL 拡張コントロールは XSL コードの中の独立した要素です。

XSL 拡張メソッドは .NET クラスのメソッドに対応します。たとえば、`sc:fld()` XSL 拡張メソッドは、`sc` 名前空間に表されている `Sitecore.Xml.Xsl.XslHelper` クラスの `fld()` メソッドに対応します。XSL 拡張メソッドは属性値の中に表われ、XML 要素としては独立していません。

一般に、XSL 拡張コントロールと比べ、XSL 拡張メソッドは書きやすく、柔軟に使うことができ、多くの機能を提供します。一方、XSL 拡張コントロールは効率的です。XSL 拡張コントロールは、より XSL に一貫したコード構文となります。

メモ
特に指定されていない限り、すべての XSL 拡張コントロールとメソッドはコンテキスト言語の各アイテムの現行のバージョンを取り扱います。

4.5 Sitecore の XSL 拡張コントロール

Sitecore は各種のレンダリング機能を簡単に使用することのできる、いくつかの .NET XSL 拡張コントロールを提供します。

4.5.1 共通の属性

この章で説明する XSL 拡張コントロールのうちのいくつかは、次のセクションで説明する共通の属性を使うことができます。

field 属性

field 属性は処理するフィールドの名前を指定します。

```
        <sc:text field="FieldName" />
```

select 属性

`select` 属性はコントロールが機能するアイテムを指定します。`select` 属性に値を指定しない場合には、コントロールはコンテキスト要素に機能します。

```
        <sc:text field="FieldName" select="$sc_item" />
```

show-title-when-blank 属性

フィールド値が空で `show-title-when-blank` 属性が `true` の場合、レイアウト エンジンはデータ テンプレート フィールドの名前をページ エディターのインライン編集コントロールの前に出力します。これは CMS ユーザーが空のフィールドを見つけるのに役立ちます。

```
        <sc:text field="FieldName" show-title-when-blank="true" select="$sc_currentitem" />
```

disable-web-editing 属性

`disable-web-editing` 属性が `true` の場合、レイアウト エンジンはページ エディターでフィールドのインライン編集を無効にします。`sc:field()` XSL 拡張メソッドに第 3 のパラメーターを渡すことによってインライン編集を無効にすることもできます。次の 2 つのステートメントは等価です。

```
        <sc:text field="FieldName" disable-web-editing="true" select="$sc_currentitem" />
        <xsl:value-of select="sc:field('FieldName', $sc currentitem, 'disable-web-editing=true')"
          disable-output-escaping="yes" />
```

恣意的な属性

`<sc:image>` と `<sc:link>` XSL 拡張コントロールは、それが生成する HTML 要素に、認識することのできない属性を渡します。

```
        <sc:image border="1" ...
        <sc:link class="ClassName" ...
```

4.5.2 Sitecore の XSL 拡張コントロール

このセクションでは個々の Sitecore XSL 拡張コントロールを説明します。

メモ

`<sc:html>`、`<sc:memo>`、`<sc:text>` の各 XSL 拡張コントロールはとても類似しています。これらの XSL 拡張コントロールの間の唯一の有意な実装上の相違は、`<sc:memo>` は `line-breaks` 属性をサポートすることです。

<sc:date> XSL 拡張コントロール

`<sc:date>` XSL 拡張コントロールは Date または Datetime フィールドの値を出力します。

`<sc:date>` XSL 拡張コントロールには次の属性が必須です：

- **`field`**: Date または Datetime フィールドの名前。`field` 属性に関する詳細は、前の「`field` 属性」のセクションを参照してください。

`<sc:date>` XSL 拡張コントロールは次の任意の属性を使うことができます。

- **`format`**: .NET の書式形式。[10]
- **`select`**: Date または Datetime フィールドを含むアイテム。select 属性に関する詳細は、前の「select 属性」のセクションを参照してください。
- **`disable-web-editing`**: インライン編集を有効または無効にします。`disable-web-editing` 属性に関する詳細は、前の「disable-web-editing 属性」のセクションを参照してください。

`sc:formatdate()` XSL 拡張メソッドは `<sc:date>` XSL 拡張コントロールと同等の機能を提供します。

```
<sc:date field ="FieldName" format="d" select="$sc currentitem" />
<xsl:value-of select="sc:formatdate(sc:fld("FieldName",$sc_currentitem),'d')" />
```

<sc:editFrame> XSL 拡張コントロール

`<sc:editFrame>` XSL 拡張コントロールは編集フレームを挿入します。[11]

メモ

この文書では`<sc:editFrame>` XSL 拡張コントロールは解説しません。

[10] .NET の日付の書式形式についての詳細は、http://msdn.microsoft.com/en-us/library/97x6twsz.aspx を参照してください。
[11] 編集フレームについての詳細は、http://sdn.sitecore.net/Reference/References%20in%20Japanese/Client%20Configuration%20Cookbook.aspx から『クライアント構成リファレンス』を参照してください。

<sc:dot> XSL 拡張コントロール

<sc:dot> XSL 拡張コントロールはページ エディターでコンテンツ マーカーを生成します。コンテンツ マーカーは Web サイトの仮想コピーを閲覧しながらのコンテンツの編集をサポートします。

<sc:dot> XSL 拡張コントロールは属性を必須としません。

<sc:dot> XSL 拡張コントロールは次の任意の属性を使うことができます。

- **select**:コンテンツ マーカーに関連付けるアイテム。select 属性に関する詳細は、前の「select 属性」のセクションを参照してください。

dot:Render() XSL 拡張メソッドは <sc:dot> XSL 拡張コントロールと同等の機能を提供します。

```
<sc:dot select="$sc_currentitem" />
<xsl:value-of select="dot:Render($sc currentitem)" />
```

重要

コンテンツ マーカーは旧式の仕様です。可能な限り、コンテンツ マーカーに代えてインライン編集を有効化してください。

<sc:html> XSL 拡張コントロール

<sc:html> XSL 拡張コントロールはフィールド タイプ Rich Text Editor または Word Document の値を出力します。

<sc:html> XSL 拡張コントロールには次の属性が必須です：

- **field**: Rich Text フィールドの名前。field 属性に関する詳細は、前の「field 属性」のセクションを参照してください。

<sc:html> XSL 拡張コントロールは次の任意の属性を使うことができます。

- **select**: Rich Text フィールドを含むアイテム。select 属性に関する詳細は、前の「select 属性」のセクションを参照してください。
- **disable-web-editing**: インライン編集を有効または無効にします。disable-web-editing 属性に関する詳細は、前の「disable-web-editing 属性」のセクションを参照してください。
- **show-title-when-blank**: フィールド値がブランクの場合フィールドのタイトルを出力する。show-title-when-blank 属性に関する詳細は、前の「field 属性」のセクションを参照してください。

sc:field() と sc:fld() XSL 拡張メソッドは <sc:html> XSL 拡張コントロールと同等の機能を提供します。
sc:field() XSL 拡張メソッドの詳細については"sc:field () XSL 拡張メソッド"のセクションを参照してください。
sc:fld() XSL 拡張メソッドの詳細については"sc:fld() XSL 拡張メソッド"のセクションを参照してください。

```
<sc:html field ="FieldName" select="$sc_currentitem" />
<xsl:value-of select="sc:field ('FieldName',$sc_currentitem)"
  disable-output-escaping="yes" />
<sc:html field ="FieldName" select="$sc currentitem" disable-web-editing="true" />
<xsl:value-of select="sc:fld('FieldName',$sc_currentitem)" />
```

<sc:image> XSL 拡張コントロール

<sc:image> XSL 拡張コントロールは、Image フィールドで参照されたイメージを使って、HTML イメージ（<img>）要素を出力します。

<sc:image> XSL 拡張コントロールには次の属性が必須です：

- **field**: Image フィールドの名前。field 属性に関する詳細は、前の「field 属性」のセクションを参照してください。

<sc:image> XSL 拡張コントロールは次の任意の属性を使うことができます。

- **select**: Image フィールドを含むアイテム。select 属性に関する詳細は、前の「select 属性」のセクションを参照してください。
- **w**: 幅（Width）、単位：ピクセル。
- **h**: 高さ（Height）、単位：ピクセル。
- **mw**: 最大幅（Maximum width）、単位：ピクセル。
- **mh**: 最大高さ（Maximum height）、単位：ピクセル。
- **la**: 言語（デフォルト：コンテキスト言語）。
- **vs**: バージョン（デフォルト：最新バージョン）。
- **db**: データベース名（デフォルト：コンテキスト データベース）。
- **bc**: 背景色（Background color、デフォルト：黒）。
- **as**: 拡大可能（Allow stretch、デフォルト：False、True にするには 1 を設定）。
- **sc**: 浮動小数点で縮小（Scale by floating point number、.25 = 25%）
- **thn**: サムネール（Thumbnail、True にするには 1 を設定）。
- **disable-web-editing**: インライン編集を有効または無効にします。disable-web-editing 属性に関する詳細は、前の「disable-web-editing 属性」のセクションを参照してください。

メモ

高さと幅に影響する <sc:image> XSL 拡張コントロールの属性は、HTML の <img> 要素の height と width 属性には対応しません。リサイズなどのイメージ操作は、サーバーからクライアントへイメージを伝送するネットワーク トラフィックを最小化するため、サーバー上で処理されます。

<sc:image> XSL 拡張コントロールは、認識できない属性をそれが生成する <img> 要素に渡します。

```
        <sc:image field ="FieldName" select="$sc_currentitem" border="1" thn="1"/>
```

sc:field() XSL 拡張メソッドは <sc:image> XSL 拡張コントロールと同等の機能を提供します。sc:field() XSL 拡張メソッドの詳細については"sc:field () XSL 拡張メソッド"のセクションを参照してください。

```
<xsl:value-of select="sc:field ('FieldName', $sc_currentitem)"
  disable-output-escaping="yes" />
```

sc:fld() XSL 拡張メソッドを使って Image フィールドの個々のプロパティにアクセスすることができます。

```
<img src="{sc:fld('FieldName', $sc currentitem, 'src')}"
  alt="{sc:fld('FieldName', $sc_currentitem, 'alt')}" />
```

<sc:link> XSL 拡張コントロール

<sc:link> XSL 拡張コントロールは HTML アンカー（<a>）要素を生成します。

<sc:link> XSL 拡張コントロールは属性を必須としません。

<sc:link> XSL 拡張コントロールは次の任意の属性を使うことができます。

- **field**: Image フィールドの名前。field 属性に関する詳細は、前の「field 属性」のセクションを参照してください。
- **select**: フィールドを含むアイテム。select 属性に関する詳細は、前の「select 属性」のセクションを参照してください。
- **text**: ユーザーが HTML <a> タグでクリックするテキスト コンテンツ。
- **disable-web-editing**: インライン編集を有効または無効にします。disable-web-editing 属性に関する詳細は、前の「disable-web-editing 属性」のセクションを参照してください。

デフォルトでは <sc:link> XSL 拡張コントロールはコンテキスト要素で表されるアイテムへのリンクを生成します。特定のアイテムへリンクするには、select 属性を使ってそのアイテムを指定します。コンテキスト要素の General Link タイプのフィールドで指定されたようにリンクするには、field 属性を使ってコントロールへのフィールドの名前を渡します。特定のアイテムの General Link フィールドで指定されたようにリンクするには、select と field 属性の両方を渡します。

text 属性または <sc:link> 要素のテキスト値を使って、リンクのテキストを指定することができます。両方とも指定した場合は、レイアウト エンジンは text 属性を無視します。たとえテキスト値が空文字列に評価されても同様です。

```
<sc:link text="click here" />
<sc:link text="this is ignored">this is output<sc:link>
```

<sc:link> XSL 拡張コントロールは、認識できない属性をそれが生成する <a> 要素に渡します。

```
<sc:link class="CSSClass" />
```

sc:fld() XSL 拡張メソッドを使って General Link フィールドの個々のプロパティにアクセスすることができます。
sc:fld() XSL 拡張メソッドの詳細については"sc:fld() XSL 拡張メソッド"のセクションを参照してください。

```
<xsl:if test="sc:fld('FieldName',$sc_currentitem,'linktype')='mailto'">
```

<sc:memo> XSL 拡張コントロール

<sc:memo> XSL 拡張コントロールは Multi-Line Text フィールドの値を出力します。

<sc:memo> XSL 拡張コントロールには次の属性が必須です：

- **field**: Miulti-Line Text フィールドの名前。field 属性に関する詳細は、前の「field 属性」のセクションを参照してください。

<sc:memo> XSL 拡張コントロールは次の任意の属性を使うことができます。

- **select**: Multi-Line Text フィールドを含むアイテム。select 属性に関する詳細は、前の「select 属性」のセクションを参照してください。
- **line-breaks**: Multi-Line Text フィールドで改行のために置き換える文字。
- **disable-web-editing**: インライン編集を有効または無効にします。disable-web-editing 属性に関する詳細は、前の「disable-web-editing 属性」のセクションを参照してください。
- **show-title-when-blank**: フィールド値がブランクの場合フィールドのタイトルを出力する。show-title-when-blank 属性に関する詳細は、前の「show-title-when-blank 属性」のセクションを参照してください。

sc:fld() と sc:field() XSL 拡張メソッドは <sc:memo> XSL 拡張コントロールと同等の機能を提供します。
sc:field() XSL 拡張メソッドの詳細については"sc:field () XSL 拡張メソッド"のセクションを参照してください。
sc:fld() XSL 拡張メソッドの詳細については"sc:fld() XSL 拡張メソッド"のセクションを参照してください。

```
<sc:memo field ="FieldName" select="$sc currentitem" />
<xsl:value-of select="sc:field ('FieldName',$sc_currentitem)"
  disable-output-escaping="yes" />
<xsl:value-of select="sc:fld('FieldName',$sc_currentitem)" />
<sc:memo field ="FieldName" select="$sc_currentitem" line-breaks="&lt;br /&gt;" />
```

<sc:sec> XSL 拡張コントロール

<sc:sec> XSL 拡張コントロールは、コンテキスト ユーザーがアイテムへの指定されたアクセス権を持っている場合に、XSL 変換エンジンにコードの囲まれている部分を実行させます。

<sc:sec> XSL 拡張コントロールは次の属性を使うことができます。

- **require**: コードのアクセス権。
- **select**: アクセス権を評価するアイテム。

require 属性は次のアクセス権をサポートします：

- **item:admin**: 管理者アクセス権。
- **item:create**: 作成アクセス権。
- **item:delete**: 削除アクセス権。

- **item:read**: 読み取りアクセス権。
- **item:rename**: 名前の変更アクセス権。
- **item:write**: 書き込みアクセス権。

たとえば：

```
<sc:sec req="item:delete" select="$sc_currentitem">
  <sc:sec req="item:create" select="$sc currentitem">
    <!--the context user has both delete and create access rights to the context item-->
  <sc:sec>
<sc:sec>
```

`sc:HasRight()` XSL 拡張メソッドは `<sc:sec>` XSL 拡張コントロールと同等の機能を提供します。

```
<xsl:if test="sc:HasRight("item:delete",$sc_currentitem)">
```

<sc:text> XSL 拡張コントロール

`<sc:text>` XSL 拡張コントロールは Single-Line Text フィールドまたは他の単純テキスト フィールドの値を出力します。

`<sc:text>` XSL 拡張コントロールには次の属性が必須です：

- **field**: フィールドの名前。`field` 属性に関する詳細は、前の「field 属性」のセクションを参照してください。

`<sc:text>` XSL 拡張コントロールは次の任意の属性を使うことができます。

- **select**: フィールドを含むアイテム。select 属性に関する詳細は、前の「select 属性」のセクションを参照してください。
- **disable-web-editing**: インライン編集を有効または無効にします。`disable-web-editing` 属性に関する詳細は、前の「disable-web-editing 属性」のセクションを参照してください。
- **show-title-when-blank**: フィールド値がブランクの場合フィールドのタイトルを出力する。show-title-when-blank 属性に関する詳細は、前の「show-title-when-blank 属性」のセクションを参照してください。
- **editormode**: ページ エディターで Word Document タイプのフィールドを含むアイテムを編集するとき、`editormode` 属性の値が `inline` である場合、エディターはポップアップ ウィンドウでなく、ページ内に埋め込まれて表示されます。
- **editorwidth**: `editormode` 属性の値が `inline` のとき、`editorwidth` の値がインライン エディターの幅をピクセル数で指定します。たとえば、`770px` などです。
- **editorheight**: `editormode` 属性の値が `inline` のとき、`editorheight` の値がインライン エディターの高さをピクセル数で指定します。たとえば、`450px` などです。

`sc:fld()` と `sc:field()` XSL 拡張メソッドは `<sc:text>` XSL 拡張コントロールと同等の機能を提供します。

```
<sc:text field ="FieldName" select="$sc_currentitem" />
<xsl:value-of select="sc:fld('FieldName', $sc currentitem)" />
<xsl:value-of select="sc:field ('FieldName', $sc_currentitem)"
  disable-output-escaping="yes" />
```

<sc:disableSecurity> XSL 拡張コントロール

<sc:disableSecurity> XSL 拡張コントロールは、囲まれている XSL コードを評価する間、XSL 変換エンジンにセキュリティ チェックを無効にさせ、そのコードを管理者ユーザーのセキュリティ コンテキストで実行させます。

<sc:disableSecurity> XSL 拡張コントロールは属性を使うことができません。

<sc:disableSecurity> XSL 拡張コントロールの詳細については、次の「<sc:enableSecurity> XSL 拡張コントロール」のセクションを参照してください。

<sc:enableSecurity> XSL 拡張コントロール

<sc:enableSecurity> XSL 拡張コントロールは、囲まれている XSL コードを評価する間、XSL 変換エンジンにセキュリティを適用させ、そのコードコンテキスト ユーザーのセキュリティ コンテキストで実行させます。

セキュリティはデフォルトで適用されるので、<sc:disableSecurity> の中以外では、<sc:enableSecurity> を使う必要はありません。たとえば：

```
<!--the system enforces security while processing this segment of code-->
<sc:disableSecurity>
  <!--the system ignores security while processing this segment of code-->
  <sc:enableSecurity>
    <!--the system enforces security while processing this segment of code-->
  </sc:enableSecurity>
  <!--the system ignores security while processing this segment of code-->
</sc:disableSecurity>
<!--the system enforces security while processing this segment of code-->
```

<sc:enableSecurity> XSL 拡張コントロールは属性を使うことができません。

sc:EnterSecurityState() と sc:ExitSecurityState() XSL 拡張メソッドは
<sc:disableSecurity> と <sc:enableSecurity> XSL 拡張コントロールと同等の機能を提供します。

```
<xsl:value-of select="sc:EnterSecurityState(false())" />
  <!--the system ignores security while processing this segment of code-->
<xsl:value-of select="sc:ExitSecurityState()" />
```

<xsl:value-of> 要素は XSL 拡張コントロールを呼び出しますが、出力を生成しません。

<sc:wordStyle> XSL 拡張コントロール

<sc:wordStyle> XSL 拡張コントロールは、Word Document タイプのフィールドのコンテンツに関連付けられた CSS スタイルを含む HTML <style>要素を生成します

<sc:wordStlye> XSL 拡張コントロールには次の属性が必須です：

- **field**: Word Document フィールドの名前。field 属性に関する詳細は、前の「field 属性」のセクションを参照してください。

<sc:wordStyle> XSL 拡張コントロールは次の任意の属性を使うことができます。

- **`select`:** Word Document フィールドを含むアイテム。select 属性に関する詳細は、前の「select 属性」のセクションを参照してください。

たとえば、**WordField** という名前の Word Document フィールドのスタイルとコンテンツをコンテキスト アイテムに埋め込むには：

```
        <sc:wordstyle field ="WordField " select="$sc_currentitem" />
        <sc:text field ="WordField " select="$sc_currentitem" />
```

4.6 Sitecore の XSL 拡張コントロール

このセクションでは XSL レンダリングで使用できるいくつかの XSL 拡張メソッドを解説します。

4.6.1 sc 名前空間：Sitecore.Data.Items.Item クラス

このセクションでは最もよく使われる Sitecore の XSL 拡張メソッドを説明します。それは `sc` 名前空間で表される `Sitecore.Xml.Xsl.XslHelper` クラスです。

重要

`Sitecore.Xml.Xsl.XslHelper` クラスはこのセクションで記載されていないいくつかの XSL 拡張メソッドを公開します。これらのメソッドの詳細については Sitecore API の文書を参照してください。[12]

sc:feedUrl() XSL 拡張メソッド

`sc:feedUrl()` XSL 拡張メソッドはアイテムの RSS URL を返します。[13] 第 1 パラメーターは RSS アイテムです。第 2 パラメーターは RSS フィードが Sitecore 認証を必要とするかどうかを示し、URL のクエリ文字列パラメーターが暗号化されたユーザー認証情報を含むかどうかを示します。

たとえば、次のような構文を使って、`$item` という名の変数で特定されたアイテムの RSS URL へのリンクを生成することができます：

```
<xsl:variable name="rssUrl" select="sc:feedUrl($item, false())" />
<xsl:if test="$rssUrl">
  <a href="{$rssUrl}">RSS</a>
</xsl:if>
```

sc:field () XSL 拡張メソッド

`sc:field()` XSL 拡張メソッドはフィールドの値を返し、ユーザーがページ エディターでインライン編集中である場合には、インライン編集をサポートするマークアップを含みます。

```
<xsl:value-of select="sc:field ('FieldName',$sc_currentitem)"
  disable-output-escaping="yes" />
```

第 3 パラメーターを使って、イメージのサイズ変更に使われるものを含め、パラメーターを渡すことができます。たとえば、`<sc:image>` XSL 拡張コントロールにサポートされる属性と等価なパラメーターを使ってイメージ フィールドを処理するためには：

[12] Sitecore API の文書については、http://sdn5.sitecore.net/Reference/Sitecore%206.aspx を参照してください。
[13] RSS に関する詳細は、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、また http://sdn.sitecore.net/End%20User/Sitecore%206%20Cookbooks.aspx から『コンテンツ編集者クックブック』を参照してください。

```
<xsl:value-of select="sc:field('FieldName', $sc_currentitem,
  'disable-web-editing=yes&amp;thn=1&amp;border=1')" disable-output-escaping="yes" />
```

sc:fld() XSL 拡張メソッド

sc:fld() XSL 拡張機能はフィールドの生の値、または XML フィールド値の中の属性の値を返します。次の例では、コンテキスト アイテムのフィールドの値を使って変数を作成します：

```
<xsl:variable name="VariableName" select="sc:fld('FieldName',$sc_currentitem)" />
```

sc:fld() のよく使われる用法は、チェックボックス フィールドが選択されたかどうかを知ることです。チェックボックス フィールドはユーザーがチェックボックスを選択した場合に値 1 を保存します。ユーザーがチェックボックスを選択したかどうかを知るためには常にこの値をチェックします。

```
<xsl:if test="sc:fld("FieldName",$sc_currentitem)!='1')">
  <!--checkbox field does not exist in context item or the user has not selected it-->
</xsl:if>
```

イメージ、ファイル、一般のリンクを含むある種のタイプのフィールドは、いくつかの属性を持った単一の XML 要素を使って値を表します。sc:fld()に第 3 パラメーターを渡して特定の属性の値を読み出すことができます。たとえば、ファイル タイプのフィールドにもとづくリンクを生成するためには、sc:fld() メソッドを使ってフィールド値から src 属性を読み出すことができます：

```
<xsl:variable name="src" select="sc:fld('FieldName',$sc_currentitem,'src')" />
<xsl:if test="$src">
  <a href="{concat('/',$src)}">
    <xsl:value-of select="concat('/',$src)" />
  </a>
</xsl:if>
```

ファイル フィールドに参照されているメディア アイテムにアクセスするには、sc:item() メソッドを使って mediaid 属性に参照されているアイテムを読み出します。

```
<xsl:variable name="mediaid" select="sc:fld('FieldName',$sc currentitem,'mediaid')" />
<xsl:if test="$mediaid">
  <xsl:variable name="mediaitem" select="sc:item($mediaid,$sc_currentitem)" />
  <xsl:if test="$mediaitem">
    <a href="{concat('/',sc:GetMediaUrl($mediaitem))}">
      <xsl:choose>
        <xsl:when test="sc:fld('title',$mediaitem)">
          <sc:text field="title" select="$mediaitem" />
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="$mediaitem/@name" />
        </xsl:otherwise>
      </xsl:choose>
    </a>
  </xsl:if>
</xsl:if>
```

重要

sc:fld() XSL 拡張メソッドは、たとえばリッチ テキスト フィールドの値を読み出すときに、わかりやすい URL を使うリンクを書き直しません。リンクを書き直すためには、次のセクション「sc:field () XSL 拡張メソッド」で記載されているように、sc:field () XSL 拡張メソッドを使用します。

sc:item() XSL 拡張メソッド

`sc:item()` XSL 拡張メソッドは、ID に対応する `<item>` 要素、または最初のパラメーターに指定された短いパスを返します。

```
<xsl:variable name="content" select="/item[@key='sitecore']/item[@key='content']" />
<xsl:variable name="content" select="sc:item('/sitecore/content',$sc_currentitem)"/>
<xsl:variable name="content"
  select="sc:item('{110D559F-DEA5-42EA-9C1C-8A5DF7E70EF9}',.)"/>
<xsl:value-of select="$content/@name" />
```

重要

参照されたアイテムを含む XML ドキュメントの要素を第 2 パラメータとして `sc:item()` 拡張メソッドに渡す必要があります。

`sc:item()` XSL 拡張メソッドが返す値をチェックすることによって、参照されたアイテムが存在しコンテキストユーザーがそれへの `item:read` アクセス権を持っているかどうかを知ることができます。たとえば：

```
<xsl:variable name="content" select="sc:item('/sitecore/content',.)"/>
<xsl:if test="$content">
  <xsl:value-of select="$content/@name" />
</xsl:if>
```

sc:path() XSL 拡張メソッド

`sc:path()` XSL 拡張メソッドはコンテンツ アイテムのわかりやすい URL を返します。

```
<a href="{sc:path($sc_currentitem)}">
```

sc:GetMediaUrl() XSL 拡張メソッド

`sc:GetMediaUrl()` XSL 拡張メソッドはメディア アイテムのわかりやすい URL を返します。

```
<xsl:variable name="mediaid" select="sc:fld('FieldName',$sc_currentitem,'mediaid')" />
<xsl:if test="$mediaid">
  <xsl:variable name="mediaitem" select="sc:item($mediaid)" />
  <xsl:if test="$mediaitem">
    <a href="{concat('/',sc:GetMediaUrl($mediaitem))}">
  </xsl:if>
</xsl:if>
```

重要

Sitecore はメディア アイテムの URL の先頭に自動的にスラッシュ文字（“/”）を置きません。これにより URL がブラウザーまたはサーバーの制限を超えてしまう場合があります。必要な場合にはメディアの URL の先頭にスラッシュ文字を置いてください。

sc:pageMode() XSL 拡張メソッド

`sc:pageMode()` XSL 拡張メソッドはプレビュー、ページ エディター、デバッガーなどのクライアント モードを示す XML 構造体を返します。これには別の機能が有効化されている場合もない場合もあります。この情報を使って別のモードの別の機能を公

開するマークアップを出力することができます。ページ モードの使用についての詳細は、『クライアント構成クックブック』を参照してください。[14]

sc:IsItemOfType() XSL 拡張メソッド

`sc:IsItemOfType()` XSL 拡張メソッドは、アイテムが特定のベース データ テンプレートから継承したデータ テンプレートをもとにしている場合に真を返します。

```
<xsl:if test="sc:IsItemOfType('basetemplate',$sc_currentitem)">
<xsl:if test="$sc currentitem/@template='basetemplate'
  or sc:IsItemOfType('basetemplate',$sc_currentitem)">
```

重要

`sc:IsItemOfType()` メソッドは特定のベース テンプレートから直接継承したアイテムには偽を返します。どちらも必要な場合には `sc:IsItemOfType()` XSL 拡張メソッドを呼び出し、テンプレート名を直接比較します。

```
<xsl:if test="sc:IsItemOfType('basetemplate',$sc currentitem)
  or $sc_currentitem/@template='basetemplate'">
```

sc:Split() XSL 拡張メソッド

`sc:Split()` XSL 拡張メソッドは選択フィールドで選択されたアイテムの ID を含む XML 構造体を返します。

`sc:Split()` XSL 拡張メソッドを使って、ツリー、マルチリスト、ツリーリスト、または Sitecore のアイテムを選択できるその他のフィールドの値を処理することができます。たとえば：

```
<xsl:for-each select="sc:Split('FieldName',$sc_currentitem)">
  <xsl:for-each select="sc:item(text(),$sc currentitem)">
    <xsl:value-of select="@name" /><br />
  </xsl:for-each>
</xsl:for-each>
```

メモ

`sc:Split()` XSL 拡張メソッドはフィールド値に含まれる ID に対応するアイテムの存在を確認しません。

sc:formatdate() XSL 拡張メソッド

`sc:formatdate()` XSL 拡張メソッドは保存されている日付の値にもとづき、Sitecore で使われている ISO フォーマットで整形された文字列を返します。日付のフォーマットに関する詳細は、前の「translate() 関数」と「`<sc:date>` XSL 拡張コントロール」のセクションを参照してください。

```
<xsl:value-of select="sc:formatdate(sc:fld('FieldName',$sc_currentitem),'d')" />
```

sc:formatdate() XSL 拡張メソッド

`sc:ToLower()` XSL 拡張メソッドは文字列の小文字の値を返します。次の条件は常に真となります：

[14] 『クライアント構成クックブック』は http://sdn5.sitecore.net/Reference/References%20in%20Japanese.aspx から参照できます。

```
<xsl:if test="sc:ToLower($sc_currentitem/@name)=$currentitem/@key">
```

重要

XPath は大文字と小文字を区別します。比較を行う前に必ず大文字と小文字の別を一貫するように変換してください。

sc:trace() XSL 拡張メソッド

`sc:trace()` XSL 拡張メソッドは Sitecore デバッガーで参照できるトレース ログにメッセージを書き込みます。たとえば：

```
<xsl:value-of select="sc:trace(concat('Context element item path:',sc:path(.)))"/>
```

この場合では、`<xsl:value-of>` 要素は出力を生成しませんが、 `sc:trace()` XSL 拡張メソッドを呼び出しトレースにメッセージを書き出します。

sc:qs() XSL 拡張メソッド

`sc:qs()` XSL 拡張メソッドは URL クエリ文字列パラメーターの値を返します。

```
<xsl:choose>
  <xsl:when
    test="sc:ToLower(sc:qs('ParameterName'))='true' or sc:qs('ParameterName')='1'">
    <!--URL query string parameter is true-->
  </xsl:when>
  <xsl:otherwise>
    <!--URL query string parameter is not true-->
  </xsl:otherwise>
</xsl:choose>
```

sc:random() XSL 拡張メソッド

`sc:random()` XSL 拡張メソッドは `System.Random.Next(int)` に返されるランダムな整数を返します。[15] たとえば、1 から１０までの間の数字を生成するには：

```
<xsl:variable name="VariableName" select="sc:random(11)" />
```

4.6.2 追加の XSL 拡張メソッド クラス

この章では XSL 拡張メソッド ライブラリとして使用することのできる、さらなるいくつかのクラスについて説明します。[16]

コンテンツ マーカーの機能を除けば、XSL レンダリングを作成するために使うデフォルトの定型コードファイルで提供されている XSL 拡張メソッドを含む唯一のクラスは、`sc` 名前空間で表される `Sitecore.Xml.Xsl.XslHelper` クラスです。このクラスは最もよく使用される XSL 拡張メソッドを含んでいます。

追加の拡張ライブラリを使用するためには、XSL レンダリングのヘッダーをそれぞれの追加の拡張ライブラリに示されているように更新してください。

---

[15] `System.Random.Next(int)` に関する詳細は、http://msdn.microsoft.com/en-us/library/system.random.next.aspx を参照してください。

[16] Sitecore API (Application Programmer Interfaces) の詳細は、http://sdn5.sitecore.net/Reference/References%20in%20Japanese.aspx を参照してください。

dateutil 名前空間:Sitecore.DateUtil

`Sitecore.DateUtil` クラスのいくつかのメソッドを XSL 拡張メソッドとして使用することができます。

```
        xmlns:dateutil="http://www.sitecore.net/dateutil"
        exclude-result-prefixes="dot sc dateutil"
```

メモ

`sc` 名前空間は日付を取り扱うために最もよく使用されるメソッドを含んでいます。

stringutil 名前空間:Sitecore.StringUtil

`Sitecore.Xml.Xsl.XslHelper` には文字列を取り扱うためいくつかのメソッドがありますが、`Sitecore.StringUtil` クラスには XSL 拡張メソッドとして使うことのできる、文字列を取り扱うためにさらに役立つメソッドがあります。

```
        xmlns:stringutil="http://www.sitecore.net/stringutil"
        exclude-result-prefixes="dot sc stringutil"
```

mainutil 名前空間:Sitecore.MainUtil

`Sitecore.MainUtil` クラスには XSL 拡張メソッドとして使うことのできる、さらにいろいろなメソッドがあります。

```
        xmlns:mainutil ="http://www.sitecore.net/mainutil"
        exclude-result-prefixes="dot sc mainutil"
```

sql 名前空間:Sitecore.Xml.Xsl.SqlHelper

`Sitecore.Xml.Xsl.Sqlhelper` クラスには XSL 拡張メソッドとして使うことができ、SQL データベースを取り扱うために役立つメソッドがあります。

```
        xmlns:sql="http://www.sitecore.net/sql"
        exclude-result-prefixes="dot sc sql"
```

Chapter 5

カスタムの XSL 拡張ライブラリ

この章ではカスタムの .NET XSL 拡張機能を実装するための技法を説明します。

カスタムの XSL 拡張機能を次のために使うことができます：

- 多くのリソースを要する操作を行う。
- 複雑な操作を行うコードの可読性を向上させる。
- 現行の Sitecore データベース以外のシステムのデータにアクセスする。

本章には次のセクションがあります。

- カスタムの XSL 拡張メソッド

5.1 カスタムの XSL 拡張メソッド

このセクションでは .NET を使ってカスタムの XSL 拡張メソッドを実装する手順について解説します。カスタムの XSL 拡張メソッドを含む自分自身の XSL 名前空間を登録するか、またはデフォルトの `sc` 名前空間にメソッドを追加することができます。

ヒント
名前空間の定義を XSL レンダリングに使う定型コード ファイルに追加することを検討します。

5.1.1 メソッドを名前空間 sc に追加する方法

メソッドを名前空間 `sc` に追加するには、`Sitecore.Xml.Xsl.XslHelper` を上書きします。

1. `Sitecore.Xml.Xsl.XslHelper` から継承するクラスを作成します。
2. `web.config` の `namespace` が `http://www.sitecore.net/sc` である `/configuration/sitecore/xslExtensions/extension` 要素で、`type` 属性の値をそのクラスの署名で置換します。名前空間 `sc` はそのクラスのメソッドを、`Sitecore.Xml.Xsl.XslHelper` ベース クラスのメソッドとあわせて公開します。

```
<extension mode="on" type="Namespace.Class,Assembly"
  namespace="http://www.sitecore.net/sc" singleInstance="true" />
```

5.1.2 XSL 拡張メソッドのライブラリ オブジェクトのプロパティへのアクセス方法

XSL 拡張クラス ライブラリのオブジェクトのプロパティにアクセスするには、明示的に `get_Property()` と `set_Property()` メソッドを使います。たとえば：

```
<xsl:if test="get_PropertyName()">
  <xsl:value-of select="set_PropertyName('PropertyValue')" />
</xsl:if>
```

この場合、XSL 要素 `<xsl:value-of>` はプロパティを設定しますが、出力は生成しません。

5.1.3 XSL 拡張メソッドの例

このセクションではカスタム .NET XSL 拡張メソッドの例を示します。

GetHome() — Sitecore.Data.Items.Item を返す

XSL レンダリングの定型コード ファイルは XPath ステートメントを使って `$home` という名前の変数を定義します。`/Sitecore/Content/Home` を開始アイテムとしていないサイトで XSL レンダリングを使う場合には、この変数はは無効です。パスをハードコードするのでなく、XSL 拡張メソッドを使ってロジックを使ってホーム アイテムを決定することができます。

まずサイトのホームアイテムを決定します。次に `Sitecore.Configuration.Factory.GetItemNavigator()` メソッドを使って `Sitecore.Data.Items.Item` を XSL レンダリングに使われる `System.Xml.XPath.XPathNodeIterator` 表現に変換します。

```
namespace Namespace.Xml.Xsl
{
  private Sitecore.Data.Items.Item GetHomeItem()
  {
    Sitecore.Data.Database db = Sitecore.Context.Database;
    Sitecore.Data.Items.Item home = database.GetItem(Sitecore.Context.Site.StartPath);
    return(home);
  }
  public class XslHelper
  {
    public Sitecore.Xml.XPath.ItemNavigator GetHome()
    {
      return(Sitecore.Configuration.Factory.CreateItemNavigator(GetHomeItem()));
    }
    public string GetHomeID()
    {
      return(GetHomeItem().ID.ToString());
    }
  }
}
```

XSL レンダリングと、新規の XSL レンダリングに使用される定型コード ファイルの`$home` 変数定義を更新します。

```
<xsl:variable name="home" select="namespace:GetHome()" />
```

メモ

一般に、XML 構造体を処理するより文字列を処理するほうがより効率的です。可能な場合は、アイテムを `System.Xml.XPath.XPathNavigator` として返すのでなく、ID を `string` として返すメソッドを使います。たとえば、すでに `$home` 変数を使っていて、その変数を定義するロジックを更新したいだけでない限り、`$home` 変数を定義するのは避けます。可能な場合は、`GetHome()` でなく、`GetHomeID()` メソッドを使用します。上述のように XSL レンダリングの定型コード ファイルを更新する場合は、この変数の宣言をコメント アウトし、不要なオーバーヘッドを避けるようにします。開発者はこの変数が必要な場合には、この行のコメントを消して元に戻すことができます。

GetRandomSiblings() ー XML を使って複数の値を返す

XSL 拡張から区切り文字または XML を使ってリストを返すことができます。この技法を使ってアイテムの ID のリストを返すことができます。これは XSL 拡張メソッド `sc:Split()` で使われるものと類似の XSL コードを使って処理できます。

たとえば、レンダリングはコンテキスト アイテムの 5 つのランダムなシブリング（兄弟）へのリンクを生成する必要があるが、コンテキスト アイテム自身に対しては必要なく、同じシブリングには 2 つのリンクは生成しないとします。次の拡張ライブラリ クラスは `Sitecore.Xml.Xsl.XslHelper` クラスから継承し、その `GetItem()` メソッドを使って `System.Xml.XPath.XPathNodeIterator` に対応する `Sitecore.Data.Items.Item` を読み出します。

```
namespace Namespace.Xml.Xsl
{
  public class XslHelper :Sitecore.Xml.Xsl.XslHelper
  {
    public XPathNodeIterator GetRandomSiblings(XPathNodeIterator iterator,int max)
    {
      Sitecore.Xml.Packet packet = new Sitecore.Xml.Packet("values","");
      iterator.MoveNext();
```

```
            Sitecore.Data.Items.Item item = GetItem(iterator);
            if(item != null )
            {
              Sitecore.Collections.ChildList children = item.Parent.Children;
              if(children.Count>1)
              {
                if(max>children.Count-1)
                {
                  max = children.Count-1;
                }
                List<Sitecore.Data.ID> ids = new List<Sitecore.Data.ID>();
                Random rand = new Random();
                while(ids.Count<max)
                {
                  int index = rand.Next(children.Count);
                  if(children[index].ID!=item.ID && !ids.Contains(children[index].ID))
                  {
                    packet.AddElement("value",children[index].ID.ToString());
                    ids.Add(children[index].ID);
                  }
                }
              }
            }
            XPathNavigator navigator = packet.XmlDocument.CreateNavigator();
            if (navigator == null)
            {
              navigator = new XmlDocument().CreateNavigator();
            }
            navigator.MoveToRoot();
            navigator.MoveToFirstChild();
            return navigator.SelectChildren(XPathNodeType.Element);
          }
        }
      }
```

このコードは下記のような XML 構造体を返します。

```
      <values>
        <value>{ID}</value>
        ...
        <value>{ID}</value>
      </values>
```

下記のようなコードを使ってこの構造体を処理することができます。

```
      <xsl:for-each select="namespace:GetRandomSiblings(.,5)">
        <xsl:for-each select="sc:item(text(),$sc_currentitem)">
          <sc:link>
            <xsl:value-of select="@name" />
            <br />
          <sc:link>
        </xsl:for-each>
      </xsl:for-each>
```

メモ

このコードは例示用のみに提供されており、少数のシブリングの処理には非効率です。