Oracle® Universal Content Management

Site Studio テクニカル・リファレンス・ガイド

10*g* リリース 3（10.1.3.3.3）

部品番号 : B51308-01

2008 年 11 月

ORACLE®

Oracle Universal Content Management Site Studio テクニカル・リファレンス・ガイド , 10*g* リリース 3 （10.1.3.3.3）

部品番号 : B51308-01

原本名 : Oracle Universal Content Management Site Studio Technical Reference Guide, 10*g* Release 3 (10.1.3.3.3)

原本著者 : Sean Cearley

原本協力者 : Brian Cheyne

目次

5 フラグメント

6 コントリビューション・リージョンと要素

7 Idoc スクリプト拡張機能

10 JSP によるサーバー側スクリプト機能

11 ASP によるサーバー側スクリプト機能

12 マネージャ設定ファイル

13 Content Tracker 統合

A サード・パーティ・ライセンス

索引

はじめに

このガイドには、Site Studio で管理される Web サイトの実装を担当する管理者に役立つ情報が含まれます。

対象読者

このドキュメントは、Site Studio で管理される組織の Web サイトを運営するシステムの実装を担当する組織内の特定のユーザーを対象としています。

ドキュメントのアクセシビリティについて

オラクル社は、障害のあるお客様にもオラクル社の製品、サービスおよびサポート・ドキュメントを簡単にご利用いただけることを目標としています。オラクル社のドキュメントには、ユーザーが障害支援技術を使用して情報を利用できる機能が組み込まれています。HTML 形式のドキュメントで用意されており、障害のあるお客様が簡単にアクセスできるようにマークアップされています。標準規格は改善されつつあります。オラクル社はドキュメントをすべてのお客様がご利用できるように、市場をリードする他の技術ベンダーと積極的に連携して技術的な問題に対応しています。オラクル社のアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト `http://www.oracle.com/accessibility/` を参照してください。

ドキュメント内のサンプル・コードのアクセシビリティについて

スクリーン・リーダーは、ドキュメント内のサンプル・コードを正確に読めない場合があります。コード表記規則では閉じ括弧だけを行に記述する必要があります。しかし JAWS は括弧だけの行を読まない場合があります。

外部 Web サイトのドキュメントのアクセシビリティについて

このドキュメントにはオラクル社およびその関連会社が所有または管理しない Web サイトへのリンクが含まれている場合があります。オラクル社およびその関連会社は、それらの Web サイトのアクセシビリティに関しての評価や言及は行っておりません。

Oracle サポート・サービスへの TTY アクセス

アメリカ国内では、Oracle サポート・サービスへ 24 時間年中無休でテキスト電話（TTY）アクセスが提供されています。TTY サポートについては、(800)446-2398 にお電話ください。アメリカ国外からの場合は、+1-407-458-2479 にお電話ください。

関連ドキュメント

詳細は、Oracle Site Studio ドキュメント・セットの次のドキュメントを参照してください（1-2 ページの「ドキュメント」も参照）。

- 『Oracle Universal Content Management Site Studio インストレーション・ガイド』
- 『Oracle Site Studio Tutorial Setup Guide』
- 『Oracle Universal Content Management Site Studio チュートリアル』
- 『Oracle Universal Content Management Site Studio コントリビュータ・ガイド』
- 『Oracle Universal Content Management Site Studio デザイナ・ガイド』
- 『Oracle Universal Content Management Site Studio マネージャ・ガイド』
- 『Oracle Site Studio リリース・ノート』

表記規則

このマニュアルでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。
イタリック体	イタリックは、ユーザーが特定の値を指定するプレースホルダ変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

サポートおよびサービス

次の各項に、各サービスに接続するための URL を記載します。

Oracle サポート・サービス

オラクル製品サポートの購入方法、および Oracle サポート・サービスへの連絡方法の詳細は、次の URL を参照してください。

```
http://www.oracle.com/lang/jp/support/index.html
```

製品マニュアル

製品のマニュアルは、次の URL にあります。

```
http://www.oracle.com/technology/global/jp/documentation/index.html
```

研修およびトレーニング

研修に関する情報とスケジュールは、次の URL で入手できます。

```
http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=3
```

その他の情報

オラクル製品やサービスに関するその他の情報については、次の URL から参照してください。

```
http://www.oracle.com/lang/jp/index.html
http://www.oracle.com/technology/global/jp/index.html
```

注意： ドキュメント内に記載されている URL や参照ドキュメントには、Oracle Corporation が提供する英語の情報も含まれています。日本語版の情報については、前述の URL を参照してください。

1

概要

『Oracle Universal Content Management Site Studio テクニカル・リファレンス・ガイド』へようこそ。このガイドでは、Site Studio の広範な技術的概要と、Site Studio を使用した Web サイトの作成方法について説明します。また、Site Studio で使用されるプロジェクト・ファイル、マーカー、タグ、サービス・コール、および Idoc スクリプトの拡張機能に関する技術情報も提供します。このガイドを最大限活用するには、HTML、JavaScript、サーバー側 Idoc スクリプトの知識があること、組織において Web マスターまたは Web 開発者の役割を果していること、および ASP または JSP の開発経験があることが必要です。

このガイドを読む前に、『Oracle Universal Content Management Site Studio デザイナ・ガイド』と『Oracle Universal Content Management Site Studio コントリビュータ・ガイド』を読んでおく必要があります。また、Site Studio を使用して 1 つ以上の Web サイトを作成している必要もあります。このガイドでは、既存のフレームワークに基づいて Web サイトを構築し、独自のニーズに応じて製品をカスタマイズできるように、Site Studio で使用されるスクリプト構文について具体的に説明します。

このガイドの構成は次のとおりです。

1.1 このガイドの表記規則

このガイド全体を通じて、次の表記規則が使用されます。

- *[CS_Instance_Dir]* という表記法は、Oracle Content Server の特定のインスタンスがインストールされているシステム上の場所を示すのに使用されます。
- スラッシュ（/）は、インターネット・アドレスの各要素を区分するのに使用されます。たとえば、`http://www.oracle.com/database/` のようになります。スラッシュは、インターネット・アドレスの最後に表記される場合とされない場合があります。
- バック・スラッシュ（\）は、サーバー、ディレクトリまたはファイルのパスのレベルを区分するのに使用されます。たとえば、*C:\Directory* のようになります。これは、Windows ファイル・システムまたは UNIX システムのファイルを示す際に適用されます。バック・スラッシュは、常にサーバー、ディレクトリまたはファイル・パスの最後に表記されます。

1.2 ドキュメント

Oracle Site Studio では、次のドキュメントを参照できます。

ドキュメント	入手できる場所
リリース・ノート	Site Studio ソフトウェアの配布パッケージに PDF ファイルとして存在します。
インストレーション・ガイド	デザイナのインストール・ディレクトリの Documentation フォルダ、および Site Studio ソフトウェアの配布パッケージに PDF ファイルとして存在します。
デザイナ・ガイド	Site Studio デザイナ・アプリケーションの「Help」メニューおよび「Help」ダイアログのボタンからアクセスできます。 ドキュメントの印刷に便利な PDF バージョンは、デザイナのインストール・ディレクトリの Documentation フォルダ、および Site Studio ソフトウェアの配布パッケージに存在します。
マネージャ・ガイド	（Content Server のユーザー・インタフェース内にある）Site Studio マネージャ・アプリケーションの「Help」リンクからアクセスできます。 ドキュメントの印刷に便利な PDF バージョンは、デザイナのインストール・ディレクトリの Documentation フォルダ、および Site Studio ソフトウェアの配布パッケージに存在します。
コントリビュータ・ガイド	Site Studio コントリビュータ・アプリケーションの「Help」リンクからアクセスできます。 ドキュメントの印刷に便利な PDF バージョンは、デザイナのインストール・ディレクトリの Documentation フォルダ、および Site Studio ソフトウェアの配布パッケージに存在します。
テクニカル・リファレンス・ガイド	デザイナのインストール・ディレクトリの Documentation フォルダ、および Site Studio ソフトウェアの配布パッケージに PDF ファイルとして存在します。
Tutorial Setup	Site Studio ソフトウェアの配布パッケージの Documentation/RVH_Tutorial フォルダに PDF ファイルとして存在します。
チュートリアル	ソフトウェアの配布パッケージの Documentation/RVH_Tutorial フォルダに PDF ファイルとして存在します。

2

Site Studio Web サイト

この章の内容は次のとおりです。

2.1 Site Studio Web サイトの構成要素

Site Studio Web サイトは、次の要素で構成されます。

- **Web サイト・プロジェクト・ファイル :** Web サイトの各セクションの階層グループ定義を含んだコンテンツ・サーバーにより管理される XML プロジェクト・ファイル。追加のサイトおよびセクション・プロパティは、カスタム・セクション・プロパティおよびサイト・アセット・カテゴリの定義とともにこのプロジェクト・ファイルに定義されます。
- **SS_GET_PAGE サービス :** 様々なパラメータを使用して、動的に生成された Web ページを戻す Web サイトのコア・サービス。Site Studio リリース 7.7 および 10*g*R3 では、このサービスは、Web サーバーのフィルタ・プラグインによって解釈されるパス・ベースのフレンドリ URL に隠蔽されています。ただし、このサービスは背後で動作を続けており、最終的な Web ページを動的に生成しています。
- **URL:** Site Studio Web サイトを実行するための URL は、標準 Web サイトの URL に似ています。Site Studio Web サイトの URL には、ドメイン名（**www.oracle.com**）、Web サイト・セクションのパス（**/products/contentserver/**）、および表示されるファイル（**index.htm**）が含まれます。これらの URL へのハイパーリンクは、Site Studio 内で様々な形式で作成できます。最終的な URL には、コンシューマ用にレンダリングされる Web ページを動的に構成するために Site Studio コンポーネントが必要とするすべての情報が含まれます。
- **プライマリ・レイアウト・ページ :** サイト階層のセクションに関連付けられたレイアウト・ページ。プライマリ・ページは、それが存在するセクションのデフォルト・ページ（ランディング・ページ）として機能し、そのセクションを宛先とするリンクのターゲットになります。
- **セカンダリ・レイアウト・ページ :** サイト階層のセクションに関連付けられたレイアウト・ページ。セカンダリ・ページは、コントリビュータによりサイトに追加される新規コンテンツ（コントリビュータ・データ・ファイルとネイティブ・ドキュメント）の背景（背景 Web ページ）として機能します。
- **コントリビューション・リージョン、要素、コントリビュータ・データ・ファイル、ssIncludeXML():** コントリビューション・リージョンには、コントリビュータ・データ・ファイル（管理対象 XML ファイル）に保存されたコンテンツを含む 1 つ以上の要素が格納されます。コンテンツは、ssIncludeXML() Idoc スクリプト拡張機能を通じて実行時に抽出されます。
- **ネイティブ・ドキュメント :** Web サイトにコンテンツを追加するためのオプションの方法。ネイティブ・ドキュメントは、（コントリビュータ・データ・ファイルをリージョンに割り当てるのと同じ方法で）コントリビューション・リージョンに割り当てるか、動的変換フラグメントを使用してページに追加できます。ドキュメントは、Dynamic Converter を使用して Web ページに変換されます。
- **フラグメント :** 単純な単一行のテキストとして、または複数のコントリビューション要素を含む高度な静的リストとして使用できる再利用可能なコンテンツの断片。フラグメントは、高度な概念であり、後の章で詳しく説明します。
- **ナビゲーション・フラグメント :** コンシューマがサイトの様々なページをナビゲートする際に使用できる方法で Web サイト階層を表現する特殊なタイプのフラグメント。ナビゲーション・フラグメントは、クライアント側スクリプトまたはサーバー側スクリプトのいずれかを使用して作成できます。
- **リスト・フラグメント :** 要素とフラグメント（特にフラグメントとして実装された要素）の両方の属性を結合する特殊なタイプのフラグメント。リスト・フラグメントを使用するより先に、様々な要素タイプ（WYSIWYG、イメージ、プレーン・テキストおよびカスタム）やより単純なフラグメントを使用する必要があります。
- **ランタイム・ファイル :** 自動生成される JavaScript ファイルと XML ファイルであり、ナビゲーション・スキームの作成やコントリビューション機能の提供のために実行時に特定のフラグメントによって使用されるナビゲーションおよびコントリビューション情報が格納されます。

このガイドの大部分では、HCSP のサーバー側スクリプト構文のみを説明します。JSP および ASP に必要とされる変更の詳細は、第 10 章「JSP によるサーバー側スクリプト機能」および第 11 章「ASP によるサーバー側スクリプト機能」を参照してください。

2.2 メタデータ・フィールド

Site Studio 製品には、Site Studio コンポーネントにより作成される次の 4 つのカスタム・メタデータ・フィールドが必要です。

- 2-3 ページの「xWebsiteObjectType」
- 2-3 ページの「xWebsiteSection」
- 2-4 ページの「xWebsites」
- 2-4 ページの「xDontShowInListsForWebsites」

2.2.1 xWebsiteObjectType

xWebsiteObjectType メタデータ・フィールドは、管理対象ドキュメントがどのタイプの Web サイト関連アイテムであるかを示すのに使用されます。このフィールドは、使用可能な管理対象オブジェクトの次の値を含むオプション・リストです。

- **Data File:** XML コントリビュータ・データ・ファイル。
- **Layout File:** HCSP、JSP または ASP のレイアウト・ページ。
- **Native Document:** ネイティブ・ドキュメント。
- **Fragment:** フラグメント・ライブラリ。
- **Image:** イメージ。
- **Script:** クライアント側スクリプト（JavaScript）。
- **Stylesheet:** CSS。
- **Project:** Site Studio Web サイト・プロジェクト・ファイル。
- **Custom Element Form:** カスタム・コントリビューション要素フォーム。
- **Validation Script:** コントリビューション要素の検証スクリプト。
- **Manager Settings:** Site Studio マネージャの構成設定ファイル。
- **Other:** その他のタイプの Web サイト・アセット。

2.2.2 xWebsiteSection

xWebsiteSection フィールドは、管理対象アイテムへのリンクが生成される際、そのリンクにターゲット・セクションが明示的に含まれていない場合に、そのアイテムを表示するのに必要な Web サイト・セクションを判断するために使用されます。このフィールドは、主にコントリビュータ・データ・ファイルとネイティブ・ドキュメントで使用されます。このフィールドには、内部的に **siteId:nodeId** 値が含まれており、Site Studio コンポーネントは標準のコンテンツ・サーバー・ページを上書きして、サイトやセクションを選択するための使いやすいユーザー・インタフェースを提供します。

このフィールドの内容は、Site Studio で使用可能な異なる URL 形式を理解する際に重要になります。詳細は、このドキュメントの後の部分で説明します。（2-10 ページの「URL 形式の比較」を参照してください。）

注意： このフィールドは、リリース 7.2 より前の Site Studio で使用されていたフォルダ・ベースの xCollectionID フィールドにかわるものです。Site Studio のインストール時に、新しい xWebsiteSection フィールドが xCollectionID フィールドの既存の値で初期化されますが、対象となるのは Web サイト関連フォルダに存在するドキュメントのみです（たとえば、以前の Site Studio Web サイトの一部であり、その Web サイトのフォルダの 1 つに格納されていたドキュメントなど）。

2.2.3 xWebsites

xWebsites フィールドは、管理対象ドキュメントの属する（コンテンツ・サーバーの）Web サイトを決定するのに使用されます。内部的には、このフィールドはサイト識別子のカンマ区切りリストです。Site Studio コンポーネントは、標準のコンテンツ・サーバー・ページを上書きして、より使いやすいサイト名のリストを提供します。

管理対象ドキュメントを含むデザイナまたはコントリビュータ・アプリケーション内でアクションを実行すると、現在のサイト識別子がその管理対象ドキュメントの xWebsites フィールドに自動的に追加されます（識別子がまだ存在しない場合）。つまり、デザイナまたはコントリビュータ・アプリケーションのサイト内で管理対象ドキュメントを使用すると、その管理対象ドキュメントは自動的にそのサイトの一部となります。

重要なのは、サイト識別子は一度追加されるとこのフィールドから自動的に削除されないということです。なぜなら、Site Studio では、現在のところ管理対象ドキュメントの参照元となる可能性のあるすべての場所を認識できないためです。設計者は、デザイナ・アプリケーション内のサイト・アセット・ペインを使用して、Web サイトを対象に管理対象アイテムを手動で追加および削除できます。

注意： xWebsites フィールドは、Site Studio リリース 7.2 より前のリリースの Site Studio で使用されていた xWebsiteID フィールドにかわるものです。Site Studio のインストール時に xWebsiteID が存在すると、その xWebsiteID フィールドの既存の値で新しい xWebsites フィールドが初期化されます。以前のリリースの Site Studio で作成されたカスタム・フラグメントとの下位互換性を維持するため、xWebsiteID フィールドは削除されず、以前と同じように動作します。

2.2.4 xDontShowInListsForWebsites

xDontShowInListsForWebsites フィールドは、サイトの動的リストに表示されないように、この管理対象ドキュメントの識別に使用された Web サイトを示すのに使用されます。このフィールドにより、動的リストでの登録と除外の機能が適切に動作します。詳細は、6-6 ページの「動的リスト」を参照してください。

2.3 パス・ベースの URL

Web サイトには、URL を通じてアクセスします。URL には、リクエストされたページをサーバーが戻すのに十分な情報が含まれる必要があります。Site Studio Web サイトの場合、URL は非常に複雑になることがあります。単一の Web ページには、通常、最終的なページを構成するために Site Studio コンポーネントがそのすべてを収集する管理対象コンテンツ・アイテムが多く含まれるためです。

SS_GET_PAGE サービス・コールは、サービス・コール・パラメータに基づいて最終的なページを動的に生成します。Site Studio リリース 7.2.1 以下では、Web サイトの URL により、このサービスが直接コールされていました。

たとえば、次のようになります。

```
http://www.company.com/idcm1/idcplg?IdcService=SS_GET_PAGE&nodeId=42
```

現在のリリースでは、Site Studio Web ページのほとんどの URL がより標準的な外観の URL を持っています。

たとえば、次のようになります。

```
http://www.company.com/products/contentserver/index.htm
```

これらの URL は、以前のサービス・コールの形式よりもずっと直感的で読みやすくなっています。また、以前の直接的な SS_GET_PAGE サービス・コールの形式よりもクロールおよび索引付けが容易です。これらの URL の使用により、わかりにくい siteId 値と nodeId 値がコンシューマから隠蔽されると同時に、設計者とコントリビュータは、以前のリリースの Site Studio よりも簡単に相対 URL を使用してブックマーク・スタイルの URL を設定できます。

SS_GET_PAGE サービス・コールは、これらの URL の背後で処理を実行するために依然として存在しますが、現在は Web サーバーのフィルタ・プラグインがすべての Site Studio 関連の URL を中間で取得して、基礎となる SS_GET_PAGE サービス・コールに渡す前にそれらを複数の構成要素に変換しています。これらのパス・ベースの URL を SS_GET_PAGE サービス・コールのパラメータに変換するロジックは、非常に複雑ですが、主に次の 3 つの手順に分けられます。

- 2-5 ページの「Web サイトの識別」
- 2-6 ページの「セクションの識別」
- 2-7 ページの「データ・ファイルまたはネイティブ・ドキュメントの識別」

2.3.1 Web サイトの識別

Web サイトを識別する最も一般的な方法は、次のようにサイトごとに一意のドメイン名を使用することです。

- http://www.acmemovies.com/
- http://www.acmetv.com/
- http://www.acmemusic.com/

別の方法として、次のようにサイトごとに一意の仮想ディレクトリを使用する方法があります。

- http://www.acme.com/movies
- http://www.acme.com/tv
- http://www.acme.com/music

Site Studio では、Site Studio Web サイトにアクセスするのにこれら 2 つの方法を使用できます。仮想ディレクトリによる方法は、常に使用可能であり、仮想ディレクトリ名として **siteId** またはルート・セクションの **urlDirName** プロパティを使用することによりデフォルトで機能します。

ドメイン名による方法も使用可能ですが、ドメイン名を設定するためのネットワーク構成と、Site Studio コンポーネントにドメイン名を指示するための Site Studio 構成が必要です。複数のドメイン名を使用して、同じ Site Studio Web サイトのアドレスを指定できます。必要であれば、そのうち 1 つのアドレスを（リダイレクト時などに使用する）デフォルト・アドレスとして選択できます。

前述のサイト・アドレス指定方法の 1 つを使用して Web サイトを参照するコンシューマは、Web サイトの各ページをナビゲートする際に、引き続き同じ方法でサイトのアドレスを指定できると想定します。これを実現するため、Site Studio では、URL 内にサイト・ドメインまたは仮想ディレクトリ名をハードコードできません。かわりに、サーバー側変数の **ssServerRelativeSiteRoot** により、現在のページへのアクセス時と同じサイト・アドレス指定方法を使用して、実行時にデータを移入します。

たとえば、次のようになります。

```
<!--$ssServerRelativeSiteRoot-->products/webservers/index.htm
```

つまり、この変数により現在のサイトのルートを基準とする相対 URL を先頭に追加して、有効なサーバー相対 URL を生成します。

2.3.2 セクションの識別

Site Studio Web サイトが識別されると、コンシューマが表示しようとしている Web サイトのセクションが URL の残り部分を使用して識別されます。これは、パスの各部分が階層内の 1 つのセクションに対応する標準の階層パスです。

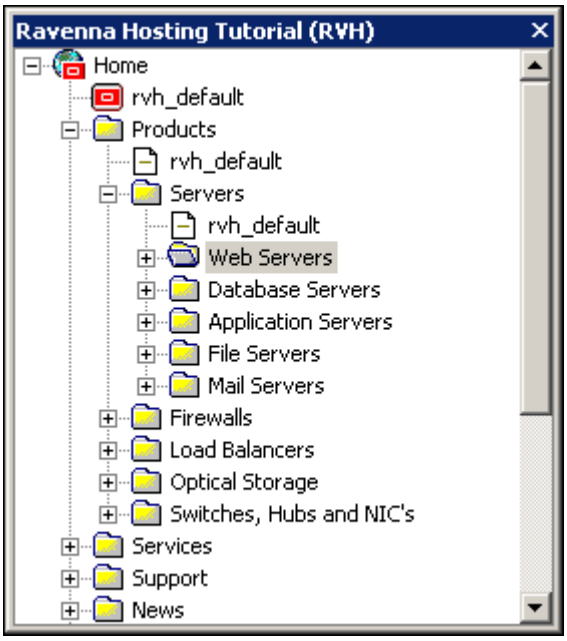
デフォルトでは、URL にセクション・ラベルが使用されますが、これは各セクションの **urlDirName** プロパティを使用して上書きし、（キャラクタ・セットが限定された最低限の長さの）URL 部分と（任意のキャラクタ・セットを使用できる任意の長さの）ラベル部分を分割することができます（また、分割する必要があります）。

たとえば、次のようになります。

```
<!--$ssServerRelativeSiteRoot-->products/servers/webservers/
```

この URL は、Ravenna Hosting Tutorial サイトの次のセクションを示しています。

図 2-1 URL が表すセクション



各要素の意味は次のとおりです。

- **products** は、Products セクションの **urlDirName** です。
- **servers** は、Servers セクションの **urlDirName** です。
- **webservers** は、Web Servers セクションの **urlDirName** です。

前述のようなセクションの URL により、そのセクションのプライマリ・ページに対するリダイレクトが発生します（これは IIS の仮想ディレクトリにおけるデフォルト・ドキュメントの設定によく似ています）。デフォルトでは、セクションのプライマリ・ページは、**index.htm** というファイルとして参照されます。この設定は、**urlPageName** セクション・プロパティの値を指定することで上書きできます。

たとえば、デフォルト値を使用した URL は、次のようになります。

```
<!--$ssServerRelativeSiteRoot>products/servers/webservers/index.htm
```

urlPageName を上書きして **overview.html** に変更した URL は、次のようになります。

```
<!--$ssServerRelativeSiteRoot>products/servers/webservers/overview.html
```

URL が Web サイト・セクションのプライマリ・ページとして識別されると、そのページ自体は次のパラメータを使用した SS_GET_PAGE コア・サービスにより構成されます。

- **nodeId:** URL のパス構成要素により識別されるセクションの一意の nodeId。

2.3.3 データ・ファイルまたはネイティブ・ドキュメントの識別

URL がセクションの urlPageName（または index.htm）と一致しないファイルで終わる場合、その URL は、URL のパス構成要素で指定されたターゲット・セクションのコンテキスト内で表示されるコントリビュータ・データ・ファイルまたはネイティブ・ドキュメントへのリンクである必要があります。URL のファイル構成要素には、表示されるデータ・ファイルまたはネイティブ・ドキュメントの dDocName が含まれます。

たとえば、次のようになります。

```
<!--$ssServerRelativeSiteRoot>products/servers/webservers/rvh_bluewebserver
```

- **products/servers/webservers/:** ターゲット・セクション・コンテキストを識別します。
- **rvh_bluewebserver:** データ・ファイルを識別します。

dDocName 値の長さには制限があるため、カスタム・メタデータ・フィールドを使用してデータ・ファイルとネイティブ・ドキュメントの URL ページ名を指定できます。この機能を有効化するには、コンテンツ・サーバーに **SSUrlFieldName** 構成エントリを追加します。

たとえば、xUrlPageName というカスタム・メタデータ・フィールドを使用する場合、次の構成エントリを追加します。

```
SSUrlFieldName=UrlPageName
```

Site Studio では、URL のファイル構成要素を使用して、最初に xUrlPageName メタデータ・フィールドにこの値を含む管理対象アイテムの検索を実行します。管理対象アイテムが見つからない場合、dDocName メタデータ・フィールドにこの値を含む管理対象アイテムの表検索を実行します。**SSUrlFieldName** 構成フラグを使用していても、dDocName は URL ページ名の構成要素として常に使用可能です。必ず一意の名前を付けることが重要です。

URL がターゲット・セクション・コンテキスト内で表示されるデータ・ファイルまたはネイティブ・ドキュメントとして識別されると、そのページ自体は次のパラメータを使用した SS_GET_PAGE コア・サービスにより構成されます。

- **ssDocName:** 表示されるデータ・ファイルまたはネイティブ・ドキュメントの dDocName
- **ssTargetNodeId:** URL のパス構成要素により識別されるセクションの一意の nodeId
- **ssTargetSiteId:** URL のドメインまたは仮想ディレクトリにより識別されるサイトの一意の siteId（これは、ssServerRelativeSiteRoot サーバー側変数により生成される場合、現在のサイトと同じになります。）

2.4 ID ベースの URL

パス・ベースの URL には、デメリットが 1 つあります。それは、リンクの作成後に階層が変更されると、破損する可能性があることです。たとえば、次のリンクがあるとします。

```
http://www.company.com/products/firewalls/index.htm
```

この形式で作成されているリンクは、次のいずれかのイベントが発生すると、破損リンクとなります。

- product セクションの urlDirName が変更された場合
- firewall セクションの urlDirName が変更された場合
- firewall セクションが新規セクションに移動された場合

基本的に、パス・ベースの URL は、階層構造の変更や、urlDirName セクション・プロパティの変更に影響されやすい性質があります。

適切な設計方針に従えば、URL がセクション・ラベルから独立するように、重要なコンテンツを作成する前にサイト階層を完成し、すべてのセクションに実用的な urlDirName プロパティの値を使用する必要があるため、これは問題となりません。

ただし、このデメリットが気になるユーザーは、サイト相対パスのかわりに nodeId 値でセクションを識別できる Site Studio の代替 ID ベース形式を使用できます。

- 2-8 ページの「サーバー側スクリプトのハイパーリンク」
- 2-9 ページの「クライアント側スクリプトのハイパーリンク」
- 2-9 ページの「URL トークンのハイパーリンク」

2.4.1 サーバー側スクリプトのハイパーリンク

ハイパーリンクは、サーバー側スクリプトのメカニズムを使用して作成し、（必要に応じて）ID ベースのパラメータを渡してセクションを識別することができます。サーバー側スクリプトを使用すると、レンダリング・ページに表示される完全に評価されたパス・ベースの URL を生成できます。これにより、コンシューマにパス・ベースの URL のメリットをすべて提供しながら、リンク作成者は ID ベースの形式を使用してリンクを指定できます。

セクションへのリンク

- **hcsp:** <!--$ssNodeLink(nodeId, siteId)-->
- **jsp:** <%=ssJSP.nodeLink(nodeId, siteId)%>
- **asp:** <%=ssNodeLink(nodeId, siteId)%>
- **xml:** [!--$ssNodeLink(nodeId, siteId)--]

nodeId の値がインスタンス全体を通じて一意である場合、siteId パラメータはオプションです。

- **hcsp:** <!--$ssLink(dDocName, nodeId, siteId)-->
- **jsp:** <%=ssJSP.link(dDocName, nodeId, siteId)%>
- **asp:** <%=ssLink(dDocName, nodeId, siteId)%>
- **xml:** [!--$ssLink(dDocName, nodeId, siteId)--]

nodeId および siteId パラメータはオプションです。nodeId の値がインスタンス全体を通じて一意である場合、nodeId パラメータが指定されていれば、siteId パラメータはオプションです。nodeId パラメータが指定されている場合、URL はルール 1（TARGET）スタイルの評価を使用して生成されます。nodeId パラメータが指定されておらず、管理対象ドキュメントの xWebsiteSection メタデータ・フィールドに値が含まれる場合、URL はルール 2（SECTION）スタイルの評価を使用して生成されます。nodeId パラメータが指定されておらず、管理対象ドキュメントの xWebsiteSection メタデータ・フィールドも空白の場合、URL は、現在のページをソース・セクション・コンテキストとして使用するルール 3（SOURCE）スタイルの評価を使用して生成されます。

2.4.2 クライアント側スクリプトのハイパーリンク

ハイパーリンクは、クライアント側スクリプトのメカニズムを使用して作成し、（必要に応じて）ID ベースのパラメータを渡してセクションを識別することができます。クライアント側スクリプトを使用しても、レンダリング・ページに表示される完全に評価されたパス・ベースの URL は生成できません。かわりに、ブラウザでリンクがクリックされると、クライアント側関数により、SS_GET_PAGE RAW サービス・コールが生成されます。この形式は、以前のリリースの Site Studio で作成されるリンクの主要形式でした（現在もデメリットはすべて同じです）。また、SS_GET_PAGE RAW サービス・コールは、最終的にパス・ベースの URL に対するリダイレクトを実行するため、コンシューマはブラウザで常にパス・ベースの URL を参照することになり、ターゲット・ページの相対リンクもすべて適切に動作します。

セクションへのリンク

- **javascript:** nodelink(nodeId, siteId);

nodeId の値がインスタンス全体を通じて一意である場合、siteId パラメータはオプションです。

javascript:nodelink() メソッドは常に SS_GET_PAGE RAW サービスにナビゲートし、nodeId 値を nodeId パラメータとして SS_GET_PAGE サービスに渡します。詳細は、2-4 ページの「パス・ベースの URL」を参照してください。

データ・ファイルまたはネイティブ・ドキュメントへのリンク

- **javascript:** link(dDocName, nodeId, siteId);

nodeId および siteId パラメータはオプションです。nodeId の値がインスタンス全体を通じて一意である場合、nodeId パラメータが指定されていれば、siteId パラメータはオプションです。nodeId パラメータが指定されている場合、javascript:link() メソッドは、その値を ssTargetNodeId パラメータとして SS_GET_PAGE サービスに渡します。siteId パラメータが指定されている場合、javascript:link() メソッドは、その値を ssTargetSiteId パラメータとして SS_GET_PAGE サービスに渡します。詳細は、2-4 ページの「パス・ベースの URL」を参照してください。

javascript:link() メソッドは常に SS_GET_PAGE RAW サービスにナビゲートし、dDocName 値を ssDocName パラメータとして SS_GET_PAGE サービスに渡します。

2.4.3 URL トークンのハイパーリンク

ハイパーリンクは、URL トークンのメカニズムを使用して作成し、（必要に応じて）ID ベースのパラメータを渡してセクションを識別することができます。URL トークンを使用すると、サーバー側スクリプトを使用する必要がなくなるため、ネイティブ・ドキュメント内でリンクを作成するのに適しています。

URL トークンを使用すると、クライアント側スクリプトも使用する必要がなくなるため、サード・パーティのインデクサがクロール可能なリンクを作成できます。ただし、URL トークンの使用には、パス・ベースの URL に対するリダイレクトも伴うため、コンシューマはブラウザで常にパス・ベースの URL を参照することになり、ターゲット・ページの相対リンクもすべて適切に動作します。

URL トークンは、Site Studio フィルタ・プラグインにより認識される特殊なキーワードであり、Site Studio URL の特別な形式として扱われます。URL トークンは、URL 内でサイト識別子（ドメイン名または仮想フォルダ名）の後の任意の場所に指定できます。フィルタ・プラグインは、トークンに続く情報を、レンダリングする Web ページを識別するためのパラメータとして使用します。

セクションへのリンク

- ssNODELINK/siteId/nodeId

nodeId の値がインスタンス全体を通じて一意である場合、siteId パラメータはオプションです。

データ・ファイルまたはネイティブ・ドキュメントへのリンク

- ssLINK/siteId/nodeId/dDocName

nodeId および siteId パラメータはオプションです。nodeId の値がインスタンス全体を通じて一意である場合、nodeId パラメータが指定されていれば、siteId パラメータはオプションです。

nodeId パラメータが指定されている場合、リダイレクト URL はルール 1（TARGET）スタイルの評価を使用して生成されます。nodeId パラメータが指定されておらず、管理対象ドキュメントの xWebsiteSection メタデータ・フィールドに値が含まれる場合、リダイレクト URL はルール 2（SECTION）スタイルの評価を使用して生成されます。nodeId パラメータが指定されておらず、管理対象ドキュメントの xWebsiteSection メタデータ・フィールドも空白の場合、リダイレクト URL は、現在のページをソース・セクション・コンテキストとして使用するルール 3（SOURCE）スタイルの評価を使用して生成されます。

2.5 URL 形式の比較

ハイパーリンクを作成する場合、リンクの形式に関しては多くの選択肢があります。形式は、どのようにリンクを構成するかを示します。リンクには、パス・ベースの URL または ID ベースの URL を選択できます。パス・ベースを選択する場合、フルパスまたは相対パスを選択できます。ID ベースを選択する場合、クライアント側スクリプト、サーバー側スクリプト、または Site Studio が提供する特殊なトークンを選択できます。各方法にはメリットとデメリットがあります。詳細は、次の比較一覧表と、それに続く各カテゴリの説明の表を参照してください。

比較一覧

メリット	フルパス	相対パス	サーバー側 ID	クライアント側 ID	URL トークン
自然なサイト・アドレス	✓	✓	✗	✗	✗
わかりにくい siteID および nodeID 値の隠蔽	✓	✓	✓	✗	✗
検索エンジンによる索引付けが可能	✓	✓	✓	✗	✓
サイト階層の変更による影響を受けない	✗	✗	✓	✓	✓
再利用されたレイアウト・ページでの動作	✓	✗	✓	✓	✓
サーバー側スクリプトが不要	✓	✓	✗	✓	✓
クライアント側スクリプトが不要	✓	✓	✓	✗	✓
サーバーでのリダイレクトが不要	✓	✓	✓	✗	✗
ブックマーク・リンクが可能	✓	✓	✗	✗	✗
パラメータの受渡しに対応	✓	✓	✗	✗	✓
ポップアップ・ウィンドウで表示可能	✓	✓	✓	✗	✓
ネイティブ・ドキュメントで使用可能	✓	✓	✗	✓	✓

各カテゴリの説明

カテゴリ	説明
自然なサイト・アドレス	URL にパス・ベースのわかりやすいアドレスが表示されます。
わかりにくい siteID および nodeID 値の隠蔽	サイトを参照するビジターとコントリビュータに、URL のわかりにくい Site Studio テクノロジが表示されません。
検索エンジンによる索引付けが可能	検索エンジンはサイトに索引を付けることができます。
サイト階層の変更による影響を受けない	リンクに影響を与えることなくサイト階層を変更できます(セクション名の変更や、あるセクションの別のセクションへの移動などが可能です)。
再利用されたレイアウト・ページでの動作	サイト全体を通じてレイアウト・ページ、データ・ファイルまたはネイティブ・ドキュメントを再利用できます。リンクは常に同じように機能します。
サーバー側スクリプトが不要	Web サイトは、サーバー側スクリプト（Idoc スクリプト、JSP、ASP など）に依存する必要がありません。
クライアント側スクリプトが不要	Web サイトは、クライアント側 JavaScript に依存する必要がありません。
サーバーでのリダイレクトが不要	Web サイトは、ID ベースのアドレスをパス・ベースのアドレスに転送するサーバー側のリダイレクトに依存する必要がありません（このリダイレクトは、サイトのパフォーマンスを低下させる可能性があります)。
ブックマーク・リンクが可能	ユーザーが Web ページの任意の場所にリンクできる、ブックマーク・リンク用のリンクを使用できます。
パラメータの受渡しに対応	適切な結果を得るために、URL にパラメータを追加する（受け渡す）ことができます。
ポップアップ・ウィンドウで表示可能	リンクを使用して、新規ポップアップ・ウィンドウを開くことができます（スクリプト処理が必要)。
ネイティブ・ドキュメントで使用可能	ネイティブ・ドキュメントで同じ形式のリンクを使用できます。

2.6 プライマリおよびセカンダリ・レイアウト・ページ

Web サイトの各セクションは、1 つのプライマリ・レイアウト・ページとオプションのセカンダリ・レイアウト・ページに関連付けられます。(プライマリおよびセカンダリ・レイアウト・ページの詳細は、『Oracle Universal Content Management Site Studio デザイナ・ガイド』を参照してください。)

各レイアウト・ページは、HTML、JavaScript およびサーバー側スクリプト（Idoc スクリプト、ASP または JSP）を格納できます。レイアウト・ページのコンテンツの大部分は、「Design」ビューでの作業時にデザイナにより自動的に生成されます。また、コンテンツは手動で「Source」ビューに追加することもできます。

このマークアップ以外に、すべてのレイアウト・ページに存在する重要な機能が 3 つあります。次の要素は、Site Studio でサイト・ナビゲーションおよびコントリビューション機能を有効化するために必要です。

- 2-12 ページの「キャラクタ・セット・エンコーディングの宣言」
- 2-12 ページの「ss_layout_head_info リソース・インクルード」
- 4-1 ページの「<ssinfo> XML データ・アイランド」

2.6.1 キャラクタ・セット・エンコーディングの宣言

レイアウト・ページの <head> セクションの最初のエントリは、レイアウト・ページで使用するキャラクタ・セット・エンコーディングの宣言である必要があります。新規レイアウトを作成すると、コンテンツ・サーバーのデフォルト・エンコーディングが使用されます。

使用するエンコーディングは、「Source」ビューでのレイアウト・ページの編集時に変更できます。次回の保存時に、レイアウトはそのキャラクタ・セット・エンコーディングを使用するように変換されます。デフォルト以外のエンコーディングに変更する場合は、現在のコンテンツ・サーバーがその指定どおりにエンコーディングされたページを提供できることを確認する必要があります。

キャラクタ・セット・エンコーディングの宣言は、<meta> タグ内に含まれます。

```
<meta http-equiv="Content-Type"
      content="text/html; charset=iso-8859-1" />
```

JSP ページの場合、次の JSP 宣言を <meta> タグより前に配置する必要があります。

```
<%@ page contentType="text/html; charset=iso-8859-1"
 pageEncoding="iso-8859-1" %>
```

ASP ページの場合、次の ASP 宣言を <meta> タグより前に配置する必要があります。

```
<% Session.CodePage=28591 %>
```

2.6.2 ss_layout_head_info リソース・インクルード

ss_layout_head_info は、サーバー側とクライアント側両方の変数およびメソッドの共通の宣言を含んだコンテンツ・サーバーの標準リソース・インクルードです。このリソース・インクルードは、レイアウト・ページの <head> セクションで、（キャラクタ・セット・エンコーディング用の <meta> タグの後に配置する）最初のエントリの 1 つとして必要であり、デザイナでレイアウト・ページを作成すると Site Studio によって自動的に生成されます。

リソース・インクルードには、レイアウト・ページで使用される言語に応じて異なる方法でアクセスします。

HCSP ページの場合 :

```
<!--$include ss_layout_head_info-->
```

JSP ページの場合 :

```
<%@include file="/WEB-INF/ss_layout_head_info.jsp"%>
```

ASP ページの場合 :

```
<!--#INCLUDE virtual="/stellent/websites/ss_layout_head_info.asp"-->
```

HCSP のリソース・インクルードのコンテンツは、Site Studio の次のコンポーネント・リソース・ファイルに存在します（<cs_name> はコンテンツ・サーバーの名前です）。

```
<cs_name>\custom\SiteStudio\resources\ss_resources.htm
```

ASP および JSP のリソース・インクルードの場所は、前述のインクルード文の構文に示されています。

ss_layout_head_info インクルードにより、レイアウト・ページに次の機能が提供されます。

- ページをコントリビューション・モードで表示する場合、次の JavaScript ファイルがインクルードされます（このモードは、Cookie または URL パラメータに SSContributor=true と設定した場合に有効化されます）。これらのファイルには、JavaScript コントリビュータ・アプリケーションの実装が含まれます。
  - <$HttpRelativeWebRoot$>resources/wcm/wcm.js
  - <$HttpRelativeWebRoot$>resources/wcm/3rdparty/dom-drag/dom-drag.js
  - <$HttpRelativeWebRoot$>resources/wcm/base/wcm.dhtml.js
  - <$HttpRelativeWebRoot$>resources/wcm/base/wcm.get.js
  - <$HttpRelativeWebRoot$>resources/wcm/base/wcm.http.js
  - <$HttpRelativeWebRoot$>resources/wcm/sitestudio/wcm.idc.js
  - <$HttpRelativeWebRoot$>resources/wcm/base/wcm.menu.js
  - <$HttpRelativeWebRoot$>resources/wcm/base/wcm.popup.js
  - <$HttpRelativeWebRoot$>resources/wcm/form/wcm.popupform.js
  - <$HttpRelativeWebRoot$>resources/wcm/base/wcm.htmlcompare.js
  - <$HttpRelativeWebRoot$>resources/wcm/base/wcm.diff.js
  - <$HttpRelativeWebRoot$>resources/wcm/sitestudio/wcm.sitestudio.popupform.js
  - <$HttpRelativeWebRoot$>resources/wcm/sitestudio/wcm.contributor.js

  次のランタイム JavaScript ファイルもインクルードされます（2-15 ページの「自動生成ランタイム・ファイル」を参照）。
  - **sitenavigationfunctions.js:** アクティブな Web サイト階層の JavaScript 表現を作成し、それにアクセスするための JavaScript メソッドが含まれます。
  - **sitenavigation.js:** アクティブな Web サイト階層の JavaScript 表現が含まれます。
- 独自の JavaScript ファイルを追加するためのカスタマイズ可能なリソースは、次のように配置されます。
  - <$include ss_extra_contributor_scripts$>

  次のスタイルシートもインクルードされます。
  - <$HttpRelativeWebRoot$>resources/wcm/sitestudio/wcm.contributor.css
  - <$HttpRelativeWebRoot$>resources/wcm/base/wcm.menu.css

  独自のスタイルシートを追加するためのカスタマイズ可能なリソースは、次のように配置されます。
  - <$include ss_extra_contributor_styles$>
- クライアント側 JavaScript でサーバー側 Idoc 変数に含まれる値にアクセスできるように、いくつかの便利な JavaScript 変数が作成されます。
  - **g_HttpRelativeWebRoot:** サーバー側 Idoc 変数 HttpRelativeWebRoot のクライアント側 JavaScript 表現が含まれます。
  - **SSContributor:** ページがコントリビューション・モードにあるかどうかを示す true または false の値が含まれます。
  - **SSHideContributorUI:** コントリビューション・モード時に表示される通常のコントリビューション UI（コントリビュータ・アイコン）を非表示にするかどうかを示す true または false の値が含まれます。これは、差分レポートの一部としてコントリビュータ・アイコンを無視するために、主にコントリビュータ・アプリケーションの差分機能で使用されます。

  - **ssUrlPrefix:** サーバー側 Idoc 変数 ssServerRelativeSiteRoot の（この例における）クライアント側 JavaScript 表現が含まれます。
  - **g_navNode_Path:** 現在のページへのパスのコンテキストを提供するセクション ID 値の配列が含まれており、ブレッドクラム・スタイルのフラグメントを実装できます。
  - **g_ssSourceNodeId:** 現在レンダリングされているページの nodeId が含まれます。
  - **g_ssSourceSiteId:** 現在レンダリングされているページの siteId が含まれます。
  - **g_strLanguageId:** コンテンツ・サーバーの現在のユーザー・プロファイルから取得された 2 文字の言語識別子が含まれます。
- **ss_page_cache_control** リソースがインクルードされ、現在のセクションの maxAge プロパティが存在する場合、このプロパティに基づいてキャッシュ制御ヘッダーが生成されます。これは、ブラウザおよびプロキシ・サーバーによる Web サイト・セクション内の各ページのキャッシングを制御するために使用されます。
- **ss_ajax_includes** リソース・インクルードでは、マネージャのファイルがサポートされます。
- ローカライズされた文字列リソースのインクルードも存在します。このインクルードでは、次のサービス・コールを使用して、現在のユーザー・ロケールに対して適切にローカライズされた文字列を取得します。
  - <$HttpCgiPath$>?IdcService=SS_GET_CONTRIBUTOR_STRINGS
- SSContributor が true ではない状態でページを表示する場合、次のインクルードのみが使用されます。
  - <$HttpRelativeWebRoot$>resources/wcm/sitestudio/wcm.consumption.js

2.7 ネイティブ・ドキュメントと ssIncInlineDynamicConversion()

Site Studio でネイティブ・ドキュメントを Web サイトに追加する場合、次の 2 つの方法があります。

- 2-14 ページの「レイアウト・ページでのインライン動的変換」
- 2-15 ページの「コントリビューション・リージョンでのインライン動的変換」

どちらの方法でも、Dynamic Converter を使用して、Web サイトで表示できるようにドキュメントを HTML に変換します。

2.7.1 レイアウト・ページでのインライン動的変換

最初の方法では、動的変換命令を使用して直接レイアウト・ページにドキュメントを配置します。デザイナでこれを行うには、「Source」ビューで適切なスクリプト拡張機能を追加するか、「Design」ビューで動的変換フラグメントを追加します。必要に応じてツールボックスで使用できる 2 つのサンプル・フラグメントがあります（『Oracle Universal Content Management Site Studio デザイナ・ガイド』を参照）。

変換では、次の 4 つのスクリプト拡張機能のいずれかを使用します。

- ssIncInlineDynamicConversion では、組込みのデフォルト・テンプレートを使用してネイティブ・ドキュメントをインライン変換します。
- ssIncInlineDynamicConversion では、指定された変換テンプレートを使用してネイティブ・ドキュメントをインライン変換します。
- ssIncDynamicConversionByRule では、名前付きの動的変換ルールを使用してネイティブ・ドキュメントをインライン変換します。
- ssIncDynamicConversionByRulesEngine では、ルール・エンジンを使用してネイティブ・ドキュメントをインライン変換します。

2.7.2 コントリビューション・リージョンでのインライン動的変換

ネイティブ・ドキュメントを変換する 2 番目の方法では、ネイティブ・ドキュメントをコントリビューション・リージョンに明示的に割り当てます。ネイティブ・ドキュメントでは、セカンダリ・レイアウト・ページの REPLACEABLE リージョンにコンテンツを提供することも可能です。この場合、ss_open_region_definition リソース・インクルードですでに説明したロジックにより、ネイティブ・ドキュメントがコントリビューション・リージョンに割り当てられていることが検出され、リージョン全体が変換済のドキュメントで置換されます。

デフォルトでは、リージョンにカスタムの動的変換コマンドが指定されていなければ（デザイナで実行可能）、ドキュメントは ssIncInlineDynamicConversion() スクリプト拡張機能を使用して変換されます（『Oracle Universal Content Management Site Studio デザイナ・ガイド』を参照）。

2.8 自動生成ランタイム・ファイル

Site Studio は、ランタイム・ファイルのコレクションを使用して、完全に動作する Web サイトを配信します。Web サイトのサイト階層を変更すると、これらのファイルが影響を受けます。これが、デザイナでサイト階層を変更したときにランタイム・ファイルの更新を求められる理由です。

ランタイム・ファイルは、Web サイトの次のランタイム・フォルダに格納されます（<cs_name> はコンテンツ・サーバーの名前、<siteid> は Web サイトです）。

```
<cs_name>\weblayout\websites\<siteid>
```

次のファイルが自動的に生成されます。

- 2-15 ページの「sitenavigation.js」
- 2-17 ページの「sitenavigationfunctions.js」
- 2-17 ページの「sitenavigation.xml」
- 2-18 ページの「sitenavigation.hda」
- 2-18 ページの「sitenavigation_co.hda」
- 2-18 ページの「wcm.consumption.js」

2.8.1 sitenavigation.js

sitenavigation.js ファイルには、Web サイト階層を定義するのに必要な JavaScript が含まれます。Site Studio の一部のナビゲーション・フラグメントは、このファイルから情報を読み取るように設計されており、クライアント側 JavaScript を使用して Web サイトのナビゲーション・スキームを動的に生成します。

sitenavigation.js には、サイト階層の単一ノード（デザイナ・インタフェースではセクションとも呼ばれる）を表す NavNode オブジェクト定義が含まれます。sitenavigation.js には、Web サイト・プロジェクト・ファイル内の実際のセクション・プロパティから生成されたプロパティがあります。詳細は、3-2 ページの「Web サイト・プロジェクト・ファイルについて」を参照してください。

ノード・プロパティ / メソッド	定義
m_parent	現在のノードの親が含まれます。
m_level	現在のノードのレベルが含まれます（0 はルート）。
m_id	現在のノードの nodeId が含まれます。
m_label	現在のノードのノード・ラベルが含まれます。
m_href	サイト階層におけるこのノードのサーバー相対パスが含まれます。
m_subNodes	子ノードの配列が含まれます。
addNode()	新しい子ノードを追加するために使用できます。

これらの標準プロパティ以外にも、NavNode オブジェクトには、このセクションの値が割り当てられたすべてのカスタム・セクション・プロパティに対応するメンバー変数が含まれます。これらの変数のネーミング規則は、cp_*XXX* です（*XXX* はカスタム・セクション・プロパティ名です）。これらのデータ・メンバーは、NavNode コンストラクタに渡された追加パラメータを解析することで構成されます。追加パラメータの形式は、名前 == 値の文字列です（後述するサンプル・コードの g_navNode_0_0 の定義を参照）。これらの追加パラメータは、ランタイム JavaScript ファイルの再生成時に Site Studio によって自動的に生成されます。

sitenavigation.js ファイルには、定式名（g_navNode_Root）を使用して単一のルート・ノードで定義される NavNode オブジェクトに関するアクティブなサイト階層の宣言も含まれます。この値は、必要に応じてナビゲーション・フラグメントで調査できます。

次に例を示します。

```
var g_navNode_Root = new NavNode('9001','Home',ssUrlPrefix +
'index.htm',null,'PageTitle==Ravenna Hosting Tutorial Site');
g_navNode_1=g_navNode_Root.addNode('9002','Products',ssUrlPrefix +
'Products/index.htm','PageTitle==Ravenna Hosting Products');
g_navNode_1_0=g_navNode_1.addNode('9003','Servers',ssUrlPrefix +
'Products/Servers/index.htm','MainNavIcon==/idcm1/groups/public/documents/
rvh_image/rvh_navicon_1.gif','PageTitle==Ravenna Hosting Servers');
g_navNode_1_0_0=g_navNode_1_0.addNode('9004','Web Servers',ssUrlPrefix +
'Products/Servers/WebServers/index.htm','MainNavIcon==/idcm1/groups/public/documents/
rvh_image/rvh_navicon_2.gif','PageTitle==Ravenna Hosting Web
Servers','SidebarProductsListBanner==webservers');
g_navNode_1_0_1=g_navNode_1_0.addNode('9005','Database Servers',ssUrlPrefix +
'Products/Servers/DatabaseServers/index.htm','MainNavIcon==/idcm1/groups/public/
documents/rvh_image/rvh_navicon_3.gif','PageTitle==Ravenna Hosting Database
Servers','SidebarProductsListBanner==databaseservers');
g_navNode_1_0_2=g_navNode_1_0.addNode('9006','Application Servers',ssUrlPrefix +
'Products/Servers/ApplicationServers/index.htm','MainNavIcon==/idcm1/groups/public/
documents/rvh_image/rvh_navicon_4.gif','PageTitle==Ravenna Hosting Application
Servers','SidebarProductsListBanner==applicationservers');
g_navNode_1_0_3=g_navNode_1_0.addNode('9007','File Servers',ssUrlPrefix +
'Products/Servers/FileServers/index.htm','MainNavIcon==/idcm1/groups/public/documents/
rvh_image/rvh_navicon_5.gif','PageTitle==Ravenna Hosting File
Servers','SidebarProductsListBanner==fileservers');
g_navNode_1_0_4=g_navNode_1_0.addNode('9008','Mail Servers',ssUrlPrefix +
'Products/Servers/MailServers/index.htm','MainNavIcon==/idcm1/groups/public/documents/
rvh_image/rvh_navicon_6.gif','PageTitle==Ravenna Hosting Mail
Servers','SidebarProductsListBanner==mailservers');
```

sitenavigation.js ファイルの NavNode 定義と JavaScript メソッドは、ランタイム・ファイルが再生成されるたびに Site Studio のコンポーネント・リソースから取得されます。NavNode オブジェクトの階層のみが動的に生成されます。他の JavaScript を変更する場合は、コンポーネント・リソースを更新または上書きする必要があります。

2.8.2 sitenavigationfunctions.js

sitenavigationfunctions.js ファイルでは、クライアント側 JavaScript のナビゲーション・メカニズムで使用されるメソッドが提供されます。たとえば、ID ベースのクライアント側ハイパーリンク関数の link() メソッドや nodelink() メソッドの定義は、どちらもこのファイルに格納されます。

2.8.3 sitenavigation.xml

sitenavigation.xml ファイルには、サーバー側スクリプトで使用できるアクティブなサイト階層の XML 定義が含まれます。このファイルで提供される情報を読み取るために、いくつかのナビゲーション・フラグメントが作成されます。各ナビゲーション・フラグメントは、この情報を読み取り、クライアント側 JavaScript ではなくサーバー側スクリプトを使用して Web サイトのナビゲーション・スキームを動的に生成します。

XML 定義には、ルートとして単一の <site> タグが含まれます。このタグには、各セクションを定義するための <section> タグの階層が含まれます。

次に例を示します。

```
<site id="9001" level="0" parent="" label="Home" href="index.htm" PageTitle="Ravenna
Hosting Tutorial Site">
<section id="9002" level="1" label="Products" href="Products/index.htm"
PageTitle="Ravenna Hosting Products">
<section id="9003" level="2" label="Servers" href="Products/Servers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_1.gif"
PageTitle="Ravenna Hosting Servers">
<section id="9004" level="3" label="Web Servers"
href="Products/Servers/WebServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_2.gif"
PageTitle="Ravenna Hosting Web Servers"
SidebarProductsListBanner="webservers"></section>
<section id="9005" level="3" label="Database Servers"
href="Products/Servers/DatabaseServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_3.gif"
PageTitle="Ravenna Hosting Database Servers"
SidebarProductsListBanner="databaseservers"></section>
<section id="9006" level="3" label="Application Servers"
href="Products/Servers/ApplicationServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_4.gif"
PageTitle="Ravenna Hosting Application Servers"
SidebarProductsListBanner="applicationservers"></section>
<section id="9007" level="3" label="File Servers"
href="Products/Servers/FileServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_5.gif"
PageTitle="Ravenna Hosting File Servers"
SidebarProductsListBanner="fileservers"></section>
<section id="9008" level="3" label="Mail Servers"
href="Products/Servers/MailServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_6.gif"
PageTitle="Ravenna Hosting Mail Servers"
SidebarProductsListBanner="mailservers"></section></section>
```

2.8.4 sitenavigation.hda

sitenavigation.hda ファイルには、ssLoadSiteNavResultSet() スクリプト拡張機能で使用できる SiteStudioNavNodes ResultSet の永続表現が含まれます。このファイルには、サーバー側スクリプトで使用できるアクティブなサイト階層の定義が含まれます。この ResultSet で提供される情報を読み取るために、いくつかのナビゲーション・フラグメントが作成されます。各ナビゲーション・フラグメントは、この情報を読み取り、クライアント側 JavaScript ではなくサーバー側スクリプトを使用して Web サイトのナビゲーション・スキームを動的に生成します。

SiteStudioNavNodes ResultSet には、次の 5 つの列があります。

- **nodeId:** ノードの一意の識別子。
- **parentNodeId:** 親ノードの一意の識別子。
- **label:** ノードのラベル。
- **level:** サイト階層におけるノードの深さ。ルート・セクションのレベルは 0（ゼロ）です。
- **href:** ノードのプライマリ・ページに対するパス・ベースのサイト相対 URL。

2.8.5 sitenavigation_co.hda

sitenavigation_co.hda ファイルには、sitenavigation.hda ファイルと同じ構造が含まれますが、SiteStudioNavNodes 結果セットにはコントリビュータ専用ノードも含まれます。

2.8.6 wcm.consumption.js

wcm.consumption.js ファイルには、Web サイトでコントリビューション機能を提供するのに必要な JavaScript が含まれます。最も重要なことは、wcm.contributor.OnKeyDown() 関数で、コントリビューション・モードに移行するためのキーボード・シーケンス（**[Ctrl]** + **[Shift]** + **[F5]**）を必要に応じて別のシーケンスに変更できることです。

3

Web サイト・プロジェクト・ファイル

この章の内容は次のとおりです。

3.1 Web サイト・プロジェクト・ファイルについて

Web サイト・プロジェクト・ファイルは、コンテンツ・サーバーで作成されるすべての Site Studio Web サイトに対して生成されます。この XML ファイルにより、Web サイト階層の定義とすべてのサイトおよびセクション・プロパティ値に加え、カスタム・セクション・プロパティおよびサイト・アセット・カテゴリの定義が提供されます。

プロジェクト・ファイルは、コンテンツ・サーバーにより管理されますが、通常、個々の設計者がチェックインまたはチェックアウトすることはありません。かわりに、Site Studio コンポーネントが、プロジェクト・ファイルのリビジョン管理を担当します。コンテンツ・サーバーが起動すると、Site Studio コンポーネントはすべてのプロジェクト・ファイルをメモリー内に読み込み、Site Studio 関連のすべてのサービスはこのメモリー内レンディションと対話します。

1 箇所が変更されただけでプロジェクト・ファイルのリビジョン変更が発生し、ファイル数が多くなることを避けるため、コンポーネントではこのメモリー内レンディションをコンテンツ・サーバー・システムに自動的にチェックインすることはありません。かわりに、コンポーネントでは、一定の期間はプロジェクト・ファイルのコピーを使用し続け、この期間における新規リビジョンのみをチェックインします。（この期間は管理者が構成できます。デフォルトは 10 分です。）

プロジェクト・ファイルの新規バージョン作成のきっかけとなるその他のイベントには、コンテンツ・サーバーの停止、Web サイトの停止、またはデザイナ・アプリケーションから起動される明示的なチェックイン・アクションがあります。プロジェクト・ファイルのメモリー内レンディションに加え、Site Studio コンポーネントは、変更イベントの発生時に常にディスク・コピーをメンテナンスします。このディスク・コピーは、コンテンツ・サーバーに予期しない障害が発生してサーバーが再起動するときに、プロジェクト・ファイルをリカバリするのに使用されます。

Web サイト・プロジェクト・ファイルのリビジョン管理を担当するのはコンポーネントであるため、ユーザーは、通常の管理対象コンテンツ・アイテムのようにファイルのチェックアウト、変更、およびチェックインを行って直接これらのファイルを変更することはできません。

最初に、Site Studio のプロジェクト・ファイルに対する制御を中断させる必要があります。これを行うには、「Site Studio Administration」ページで実行中の Web サイトを停止します。これで、プロジェクト・ファイルをチェックアウトして、通常の管理対象コンテンツ・アイテムのように変更することが可能になります。手動による変更が完了したら、必ず「Site Studio Administration」ページで Web サイトを再起動する必要があります。

サイトの停止中は、実行中の Web サイトを使用できないことに注意してください。スタンドアロン HCSP ページを「Stopped Site Page」サイト・プロパティに割り当てて、コンシューマにサイトが一時的に使用できないことを通知する必要があります。

Web サイト・プロジェクト・ファイルでは、XML スキーマ定義ファイル（XSD）に対して検証される XML 構文を使用します。プロジェクト・ファイル構文の完全な説明は、Site Studio インストール・ディレクトリの次の場所にある XSD ファイルで参照できます。

```
<CS_install>\custom\SiteStudio\support\ss_project_schema.xsd
```

この構文の詳細は、3-3 ページの「プロジェクト・ファイルの構造」を参照してください。

ただし、<section> プロパティの 2 つのパラメータ（プライマリおよびセカンダリ URL リージョン宣言パラメータ）については、追加の説明が必要です。

3.2 プライマリおよびセカンダリ URL リージョン宣言パラメータ

Web サイト・プロジェクト・ファイル内の <section> タグの 2 つの属性（primaryUrl と secondaryUrl）は、Web サイトにとって非常に重要です。これらの属性により、管理対象レイアウト・ファイルと、そのレイアウト・ファイルに含まれる各コントリビューション・リージョンで使用するコントリビュータ・データ・ファイルを指定します。

たとえば、次の値を持つ primaryUrl があるとします。

```
rvh_productsprimary?region1=rvh_webservers&region2=rvh_logocontent
```

この値は、このセクションのプライマリ・レイアウトの dDocName が rvh_productsprimary であることと、レイアウトに region1 と region2 という識別子を持つ 2 つのコントリビューション・リージョンが含まれることを示しています。また、各リージョンで使用するコントリビュータ・データ・ファイルも指定しています。

プライマリおよびセカンダリ URL リージョン宣言パラメータは、デザイナにより生成されます。より具体的には、これらのパラメータは、サイト階層の特定のセクションにレイアウト・ページを作成し、「Assign Region Content」ダイアログ・ボックスを使用してレイアウト・ページ内の各リージョンにコントリビュータ・データ・ファイルを割り当てたときに生成されます。

primaryUrl および secondaryUrl セクション・プロパティに加え、secondaryUrlVariableField と呼ばれる特殊なプロパティがあります。このプロパティは、REPLACEABLE リージョンとしてマークされた secondaryUrl 内のコントリビューション・リージョンの名前を含んでいるため、このセクションに格納された任意のコントリビュータ・データ・ファイルまたはネイティブ・ドキュメントのコンテンツで置換することが可能です。

3.3 プロジェクト・ファイルの構造

プロジェクト・ファイルは、Site Studio インストール・ディレクトリの次の場所にある XML スキーマ定義ファイルに対して検証されます。

```
<CS_Install>\custom\SiteStudio\support\ss_project_schema.xsd
```

コーディングの観点からは、プロジェクト・ファイルは次のように表現されます。

```
<project>
    <section/>
    <section/>
    <section/>
         <section/>
    <section/>
    <customPropertyDefinitions>
        <customProp/>
        <customProp/>
        <customProp/>
    </customPropertyDefinitions>
    <assetCategories>
        <assetCategory>
        <assetCategory>
    </assetCategories>
    <environmentProperties>
        <environmentProperty>
        <environmentProperty>
    </environmentProperties>
</project>
```

プロジェクト・ファイルの構造内には、次のタグがあります。

3.4 <project> タグ

<project> タグは、コンテンツ・サーバーでホストされる単一の Web サイトを表します。このタグの属性は次のとおりです。

- **siteId:** この Web サイトの一意の識別子。
- **siteLabel:** この Web サイトのわかりやすいラベル。
- **siteType:** サイト・タイプ。次の値のいずれかです。
  - **idoc:** .hcsp および .jsp スタイルのレイアウトを格納できる Web サイト
  - **asp:** .asp スタイルのレイアウトのみを格納できる Web サイト
- **siteLinkFormat:** この Web サイト内にハイパーリンクを作成する際に使用するデフォルト形式。次の値のいずれかを含むことができます。
  - **FullFriendlyUrl:** 絶対パス・ベースの形式を使用します。
  - **RelativeFriendlyUrl:** 相対パス・ベースの形式を使用します。
  - **ClientSideScript:** クライアント側の JavaScript ベースの形式を使用します。
  - **ServerSideScript:** サーバー側の idoc、asp または jsp ベースの形式（レイアウト・タイプに依存）を使用します。
  - **UrlToken:** リダイレクト URL トークン・ベースの形式（非推奨）を使用します。
- **siteShowAccessibilityMenus:** true の場合、コントリビュータでメニューを有効化できます。
- **siteLinkFormatIncludeSiteId:** true の場合、ID ベースの形式のいずれかを使用する際にオプションの siteId パラメータを含めることができます。
- **siteLinkFormatHideInContributor:** true の場合、コントリビュータはハイパーリンク・ウィザードでリンク形式を選択できません。false の場合、コントリビュータはリンクを作成するたびにリンク形式を選択できます。
- **siteLinkFormatHideInDesigner:** true の場合、設計者はハイパーリンク・ウィザードでリンク形式を選択できません。false の場合、設計者はリンクを作成するたびにリンク形式を選択できます。
- **errorNodeId:** エラー・ハンドラとして使用されるセクションの識別子が含まれます。Web サイトで予期しない問題が発生した場合、この識別子を使用してエラー・ページが表示されます。
- **stoppedStateDocName:** Web サイトが停止したときにコンシューマに表示されるスタンドアロン HCSP ページの dDocName が含まれます。
- **dSecurityGroup**、**dDocAccount**、**xClbraUser**、**xClbraAlias:** サイト・プロパティへのアクセス権を制御する Web サイトのセキュリティ設定が含まれます。
- **format:** このプロジェクト・ファイルの生成に使用された Site Studio のリリースを示す内部識別子。
- **nextNodeId:** この Web サイトで作成される次のセクションに使用される数値が含まれます。この値はオプションです。Site Studio では、プロジェクト・ファイルにこの属性の値が含まれない場合に使用できる、インスタンス全体に有効な値を内部的に管理しています。

- **originalCollectionId:** このサイトが、以前のリリースの Site Studio（リリース 7.2.1 以下）で作成された既存のサイトからアップグレードされている場合、元の Web サイトの xCollectionId 値が含まれます。
- **originalInstance:** このサイトが、以前のリリースの Site Studio（リリース 7.2.1 以下）で作成された既存のサイトからアップグレードされている場合、コンテンツ・サーバーの元のインスタンス名が含まれます。
- **xmlns**、**xmlns:xsi**、**xsi:schemaLocation:** 名前空間と検証スキーマ用の標準の XML 宣言。

これらの標準の属性以外にも、<project> タグには、Web サイトに定義された任意のカスタム・サイト・プロパティの名前 / 値ペアが含まれます。

現在のところ、これらの値を作成するための GUI はありませんが、手動で入力して、任意のサイト・プロパティ関連のサービス（SS_GET_SITE_PROPERTY など）を使用することで各値にアクセスできます。

<project> タグには、次の子タグが含まれます。

- 3-5 ページの「<section> タグ」
- 3-6 ページの「<customPropertyDefinitions> タグ」
- 3-7 ページの「<assetCategories> タグ」
- 3-7 ページの「<environmentProperties> タグ」

3.5 <section> タグ

<section> タグは、Web サイト内の単一のセクションを表します。このタグの属性は次のとおりです。

- **nodeId:** Web サイトのこのセクションに対応する一意の識別子。
- **active:** true または false に設定し、このセクションを Web サイトでアクティブにするかどうかを指定します。
- **contributorOnly:** true または false に設定し、このセクションをコントリビューション・モードでのみアクティブにするかどうかを指定します。
- **label:** ナビゲーション・フラグメントにより表示される際に、Web サイト階層のこのセクションで使用されるラベル。
- **urlDirName:** Web サイト階層のこのセクションでパス・ベースの URL に使用される値。
- **urlPageName:** Web サイト階層のこのセクションのプライマリ・ページでパス・ベースの URL に使用される値（指定されない場合、デフォルトで index.htm になります）。
- **primaryUrl:** Web サイトのこのセクションに対応する相対プライマリ URL（3-3 ページの「プライマリおよびセカンダリ URL リージョン宣言パラメータ」を参照）。
- **secondaryUrl:** Web サイトのこのセクションに対応する相対セカンダリ URL（3-3 ページの「プライマリおよびセカンダリ URL リージョン宣言パラメータ」を参照）。
- **secondaryUrlVariableField:** REPLACEABLE とみなされるこのセクションに対応するセカンダリ・レイアウトのリージョンの名前（3-3 ページの「プライマリおよびセカンダリ URL リージョン宣言パラメータ」を参照）。
- **maxAge:** Web サイトのこのセクション内のページをブラウザやプロキシ・サーバーでキャッシュする秒数の値。
- **maxAgeSecondary:** Web サイトのこのセクション内のセカンダリ・ページをブラウザやプロキシ・サーバーでキャッシュする秒数の値。
- **dSecurityGroup**、**dDocAccount**、**xClbraUser**、**xClbraAlias:** セクション・プロパティへのアクセス権を制御する Web サイトのこのセクションのセキュリティ設定が含まれます。

- **navLabel:** リリース 7.2.1 以下からアップグレードされたサイトとの下位互換性を維持するために提供されています。ルート・セクションにのみ適用されます。ナビゲーション・フラグメントにより表示される際に、ルート・セクションで使用されるラベルが含まれます（これがラベル値と異なる必要がある場合）。
- **siteType:** リリース 7.2.1 以下からアップグレードされたサイトとの下位互換性を維持するために提供されています。サイト・タイプが含まれます。
- **errorNodeId:** リリース 7.2.1 以下からアップグレードされたサイトとの下位互換性を維持するために提供されています。エラー・ハンドラとして使用されるセクションの識別子が含まれており、実行中の Web サイトで予期しない問題が発生した場合、この識別子を使用してエラー・ページが表示されます。

これらの標準の属性以外にも、<section> タグには、Web サイトのこのセクションに定義された任意のカスタム・セクション・プロパティの名前 / 値ペアが含まれます。

<section> タグには、その子セクションを表す 0（ゼロ）個以上の <section> タグが含まれます。

3.6 <customPropertyDefinitions> タグ

<customPropertyDefinitions> タグは、Web サイト内のカスタム・セクション・プロパティ定義のグループを表します。このタグは、属性を持ちませんが、0（ゼロ）個以上の <customProp> タグを格納できます。

<customProp>

<customProp> タグは、単一のカスタム・セクション・プロパティ定義を表します。このタグの属性は次のとおりです。

- **name:** このカスタム・セクション・プロパティの名前。
- **description:** このカスタム・セクション・プロパティの説明。
- **type:** このカスタム・セクション・プロパティのタイプ。次のいずれかの値である必要があります。
  - **text:** 単純なテキスト型パラメータ。
  - **bigtext:** 複数行のテキスト型パラメータ。
  - **boolean:** 単純な True/False 型パラメータ。
  - **integer:** 単純な整数型パラメータ。
  - **float:** 単純な浮動小数点型パラメータ。
  - **size:** HTML サイズ（10px、50%、3pt など）を表すパラメータ。
  - **color:** 色の値（RGB または HTML の色名を使用）。
  - **url:** URL。
  - **manageddoc:** 「Search Results」ページで選択された、Content Server の管理対象ドキュメントの dDocName。
  - **managedurl:** 「Search Results」ページで選択された、Content Server の管理対象ドキュメントの DocUrl。
  - **managedquery:** 「CAPTURE_QUERY」ページで選択された問合せテキスト文字列。
  - **cssstyle:** CSS スタイル属性を表すテキスト・パラメータ。このタイプは、text タイプのように動作します。（将来的な使用を意図しています。）
  - **siteid:** 「Select Site」ダイアログ・ボックスで選択された、コンテンツ・サーバーの同じインスタンスに含まれるいずれかの Web サイトの siteId 値。
  - **nodeid:** 「Select Section」ダイアログ・ボックスのサイト階層を通じて選択された、Web サイトのいずれかのセクションの nodeId 値。

<customProp> タグには、子タグは含まれません。

3.7 <assetCategories> タグ

<assetCategories> タグは、Web サイト内のアセット・カテゴリ定義のグループを表します。このタグは、属性を持ちませんが、0（ゼロ）個以上の <assetCategory> タグを格納できます。

<assetCategory>

<assetCategory> タグは、単一のアセット・カテゴリ定義を表します。このタグの属性は次のとおりです。

- **name:** このサイト・アセット・カテゴリの名前
- **description:** このサイト・アセット・カテゴリの説明
- **type:** このサイト・アセット・カテゴリの有効な xWebsiteObjectType 値
- **querytext:** このサイト・アセット・カテゴリを構成するドキュメントを定義するための問合せテキスト文字列
- **metadata:** このサイト・アセット・カテゴリ用のデザイナ・アプリケーションのサイト・アセット・ペインを通じて作成されたドキュメントのデフォルト・メタデータ定義文字列

<assetCategory> タグには、子タグは含まれません。

3.8 <environmentProperties> タグ

<environmentProperties> タグは、サーバーの環境定義のグループを表します。このタグは、属性を持ちませんが、0（ゼロ）個以上の <environmentProperty> タグを格納できます。

<environmentProperty>

<environmentProperty> タグは、単一の環境カテゴリ定義を表します。このタグの属性は次のとおりです。

- **maxAge:** Web サイトのこのセクション内のページをブラウザやプロキシ・サーバーでキャッシュする秒数の値
- **maxAgeSecondary:** Web サイトのこのセクション内のセカンダリ・ページをブラウザやプロキシ・サーバーでキャッシュする秒数の値
- **stoppedStateDocName:** サーバーの停止時に表示される Site Studio のレイアウト・ページのコンテンツ ID

<environmentProperty> タグには、子タグは含まれません。

4

<ssinfo> XML データ・アイランド

この章の内容は次のとおりです。

4.1 <ssinfo> XML データ・アイランドについて

<ssinfo> XML データ・アイランドでは、Site Studio でコントリビュータ・データ・ファイルを作成してレイアウト・ページに追加できるように、コントリビューション・リージョン、要素およびフラグメントに関する構造情報が提供されます。この XML データ・アイランドは、Site Studio によって管理されており、（デザイナで）コントリビューション・リージョン、要素およびフラグメントを追加または編集すると必ず更新されます。

レイアウト・ページの <head> セクションに存在する <ssinfo> XML データ・アイランドでは、ルートに <ssinfo> タグを使用します。データ・アイランドには、レイアウト・ページで使用されるすべてのリージョン、要素およびフラグメントの詳細構造が含まれます。データ・アイランドの主な用途は、コントリビュータ・アプリケーションに構造情報を提供することです。データ・アイランドには、実行中の Web ページをコンシューマが表示したときに、そのページのフラグメントで使用されるサーバー側スクリプトの様々なフラグメント・パラメータ変数も含まれます。

コーディングの観点からは、データ・アイランドは次のように表現されます。

```
<script id="ssInfo" type="text/xml" warning="DO NOT MODIFY!">
    <ssinfo>
        <region>
            <element>
                <description>Element description goes here</description>
            </element>
            <element>
                . . .
            </element>
        </region>
        <fragmentinstance>
        </fragmentinstance>
    </ssinfo>
</script>
```

<ssinfo> XML データ・アイランド内には、次のタグがあります。

- 4-2 ページの「<script> タグ」
- 4-3 ページの「<ssinfo> タグ」
- 4-3 ページの「<region> タグ」
- 4-6 ページの「<element> タグ」
- 4-16 ページの「<fragmentinstance> タグ」

4.2 <script> タグ

<script> タグは、HTML コンテンツとは対照的な XML コンテンツ（XML データ・アイランド）を示すのに使用されます。このタグには、次の属性が含まれます。

- **id:** <ssinfo> XML データ・アイランドの一意の識別子。値は "ssInfo" に設定する必要があります。
- **type:** このタグに含まれるスクリプトのタイプ。XML を示すには、この属性の値を "text/xml" に設定する必要があります。
- **warning:** このカスタム属性は、Site Studio により追加され、"DO NOT MODIFY" という値を含みます。この値により、設計者は、デザイナの「Source」ビューで XML コードを変更する際に特に注意を払うよう促されます。

<script> タグには、<ssinfo> という単一の子タグが含まれます（4-3 ページの「<ssinfo> タグ」を参照）。

注意： <script> タグは、以前のリリースの Site Studio で使用されていた <xml> タグにかわるものです。<script> タグは複数のブラウザに対応していますが、<xml> タグは Internet Explorer のみに対応しています。

4.3 <ssinfo> タグ

<ssinfo> タグは、XML データ・アイランドのルートです。このタグは、属性を持ちませんが、次のタグを格納できます。

- 0（ゼロ）個以上の <region> タグ（4-3 ページの「<region> タグ」を参照）
- 0（ゼロ）個以上の <fragmentinstance> タグ（4-16 ページの「<fragmentinstance> タグ」を参照）

4.4 <region> タグ

<region> タグは、レイアウト・ページの単一のコントリビューション・リージョンを表します。このタグの属性は次のとおりです。

- **id:** レイアウト・ページのこのリージョンに対応する一意の識別子。
- **name:** レイアウト・ページのこのリージョンに対応する表示名。同時に、コントリビュータ・データ・ファイル内のリージョンに対応するルート XML タグの名前。
- **flags:** 9 つのブール型フラグ（1 または 0）を含む文字列値。特定のコントリビューション機能（次の表を参照）を有効化または無効化します。111000100 の値は、更新、プレビュー、リセットおよびドキュメント情報の各機能を有効化します。

フラグ	説明
0	更新機能を有効化します。
1	プレビュー機能を有効化します。
2	リセット機能を有効化します。
3	コントリビュータが更新時にメタデータを変更できます。
4	承認機能を有効化します。
5	却下機能を有効化します。
6	ドキュメント情報機能を有効化します。
7	データ・ファイル切替え機能を有効化します。
8	使用状況レポート表示機能を有効化します。
9	コンテンツ追跡レポート機能を有効化します。
'0	メタデータ更新機能を有効化します。

- **metadata:** 事前定義されたメタデータ値を含む URL エンコード済の文字列。これは、コントリビュータが更新時にメタデータ値を変更できる場合でも、特定のメタデータ・フィールドを隠蔽するために使用されます（flags 属性のフラグ 3 が 1 に設定されている場合にのみ適用されます）。コンテンツ・サーバー・ページの isHidden 機能が使用されます。たとえば、次のようになります。

  `dDocType:isHidden=true`
  （URL エンコードされた文字列は、`dDocType%3AisHidden%3Dtrue` になります。）

- **dccommand:** XML データ・ファイルのかわりにネイティブ・ドキュメントがこのリージョン内に表示されるときに、このリージョンで動的変換を実行するための Idoc コマンド文字列。

  この文字列属性の一般的な値は、次のとおりです。

  ```
  ssIncInlineDynamicConversion(SS_DATAFILE)
  ssIncDynamicConversion(SS_DATAFILE, 'ceremonial', 'snippet_layout')
  ssIncDynamicConversionByRule(SS_DATAFILE, 'My Rule')
  ssIncDynamicConversionByRulesEngine(SS_DATAFILE)
  (設計者が定義した任意のカスタム Idoc 文字列)
  ```

<region> タグは、1 つの <switchregioncontent> タグ（4-5 ページの「<switchregioncontent> タグ」を参照）と 0（ゼロ）個以上の <element> タグ（4-6 ページの「<element> タグ」を参照）を格納できます。

また、このタグには、2 つのサーバー側変数を宣言する Idoc スクリプト・タグ（HTML コメント・タグ内に配置）が含まれます。変数の 1 つには、コントリビューション・アイコンの右クリック・ポップアップ・メニューで使用できるアクションが含まれます（コントリビューション・モードでページを表示すると、各コントリビューション・リージョンにコントリビューション・アイコンが表示されます）。もう 1 つの変数には、動的変換コマンド文字列（前述の dccommand 属性）のコピーが含まれます。変数名は、リージョン識別子と、それに続く _ACTIONS または _DCCOMMAND で構成されます。

regionX_ACTIONS 変数の値は、許可されるアクションに応じて次の文字を 0（ゼロ）個以上含む文字列です。

アクション	定義
E	このリージョンで編集機能が許可されます。コントリビュータまたは Check Out and Open アプレットを起動します。
C	このリージョンで Check Out and Open 機能が許可されます。
I	このリージョンでドキュメント情報機能が許可されます。
M	このリージョンでドキュメント情報更新機能が許可されます。
P	このリージョンでワークフロー承認機能が許可されます。
R	このリージョンでワークフロー却下機能が許可されます。
S	このリージョンでデータ・ファイル切替え機能が許可されます。
U	このリージョンで使用状況レポート表示機能が許可されます。
T	このリージョンでコンテンツ追跡機能が許可されます。
D	このリージョンで差分機能が許可されます。

regionX_DCCOMMAND 変数の値は、dccommand 属性のコピーです。

たとえば、次のように使用できます。

```
<!--$region1_ACTIONS="EIPR",
     region1_DCCOMMAND="ssIncInlineDynamicConversion(SS_DATAFILE)" -->
```

これらの Idoc 変数は、flags 属性と dccommand 属性に格納された情報の複製ですが、これらの変数によりサーバー側スクリプトに同じ情報が公開されます。この情報を使用して、コントリビューション・アイコンの右クリック・ポップアップ・メニューが動的に生成され、リージョン・コンテンツの必要な動的変換が実行されます。

4.5 <switchregioncontent> タグ

<switchregioncontent> タグは、リージョンに割り当てられたコントリビューション・リージョン・ファイルを切り替える際に、コントリビュータが実行できるアクションを表します。このタグの属性は次のとおりです。

- **createnewxml:** true または false に設定し、リージョンに割り当てられる新規データ・ファイルの作成をコントリビュータに許可するかどうかを指定します。
- **createnewnative:** true または false に設定し、リージョンに割り当てられる新規ネイティブ・ドキュメントの作成をコントリビュータに許可するかどうかを指定します。
- **choosemanaged:** true または false に設定し、リージョンに割り当てられる既存の管理対象アイテムの選択をコントリビュータに許可するかどうかを指定します。
- **chooselocal:** true または false に設定し、新規管理対象アイテムとしてチェックインされ、リージョンに割り当てられる既存のローカル・ファイル・システム・ベースのアイテムの選択をコントリビュータに許可するかどうかを指定します。
- **choosenone:** true または false に設定し、リージョンに対する関連付けの削除をコントリビュータに許可するかどうかを指定します。

<switchregioncontent> タグは、次のタグを格納できます。

- <createnewnativedoctypes>（4-5 ページの「<createnewnativedoctypes>」を参照）
- <choosemanagedquerytext>（4-5 ページの「<choosemanagedquerytext>」を参照）
- <defaultmetadata>（4-6 ページの「<defaultmetadata>」を参照）

4.5.1 <createnewnativedoctypes>

<createnewnativedoctypes> タグは、リージョンに割り当てられたコントリビューション・リージョン・ファイルを切り替える際に、コントリビュータが作成できるネイティブ・ドキュメントのタイプを表します。このタグは、属性を持ちませんが、ドキュメント拡張子のカンマ区切りリストを含む単一の CDATA 子セクションを格納する必要があります。たとえば、次のようになります。

```
<createnewnativedoctypes>
   <![CDATA[.doc,.ppt,.txt,.xls]]>
</createnewnativedoctypes>
```

4.5.2 <choosemanagedquerytext>

<choosemanagedquerytext> タグは、リージョンに割り当てられる既存の管理対象アイテムの選択をコントリビュータに許可する場合に使用する問合せ文字列を表します。このタグの属性は次のとおりです。

- **corecontentonly:** true の場合、検索結果はこの問合せ文字列のみに制限されます。false の場合、コントリビュータはカスタム検索を実行できます。

<choosemanagedquerytext> タグは、問合せテキスト文字列を含む単一の CDATA 子セクションを格納する必要があります。たとえば、次のようになります。

```
<choosemanagedquerytext>
   <![CDATA[dDocType <matches> 'ADACCT']]>
</choosemanagedquerytext>
```

4.5.3 <defaultmetadata>

<defaultmetadata> タグは、リージョンに割り当てられる新規管理対象アイテムの作成をコントリビュータに許可する場合に使用するデフォルト・メタデータを表します。このタグは、属性を持ちませんが、URL エンコード済のデフォルト・メタデータ文字列を含む単一の CDATA 子セクションを格納する必要があります。たとえば、次のようになります。

```
<defaultmetadata>
   <![CDATA[dDocType%3DADACCT%26dDocType%3AisHidden%3Dtrue]]>
</defaultmetadata>
```

4.6 <element> タグ

<element> タグは、レイアウト・ページのコントリビューション・リージョンの単一の要素を表します。このタグの属性は次のとおりです。

- **id:** レイアウト・ページのコントリビューション・リージョンのこの要素に対応する一意の識別子。
- **name:** コントリビュータ・データ・ファイルにこの要素のコンテンツを格納するために使用する XML タグの名前。
- **label:** デザイナおよびコントリビュータで使用するこの要素の表示名。
- **type:** 要素タイプ。次の 6 つのタイプがあります。
  1. WYSIWYG（4-8 ページの「要素タイプ : WYSIWYG」を参照）
  2. 静的リスト（4-9 ページの「要素タイプ : 静的リスト」を参照）
  3. 動的リスト（4-9 ページの「要素タイプ : 動的リスト」を参照）
  4. カスタム（4-9 ページの「要素タイプ : カスタム」を参照）
  5. イメージ（4-10 ページの「要素タイプ : イメージ」を参照）
  6. プレーン・テキスト（4-10 ページの「要素タイプ : プレーン・テキスト」を参照）
- **fragmentid:** この要素を実装したフラグメントの一意の識別子（タイプ 2 または 3 に適用）。
- **library:** この要素を実装したフラグメントを含むフラグメント・ライブラリの一意の識別子（タイプ 2 または 3 に適用）。
- **flags:** 多くのブール型フラグ（1 または 0）を含む文字列値。次の表に記載された特定のコントリビューション機能を有効化または無効化します（53 が一番右のフラグであり、0 が一番左のフラグです）。

フラグ	説明	適用される要素タイプ
0	元に戻す	1、2、5、6
1	再実行	1、2、5、6
2	切取り	1、5、6
3	コピー	1、5、6
4	貼付け	1、5、6
5	太字	1
6	イタリック	1
7	下線	1
8	書式の削除	1
9	番号付きリスト	1
10	記号付きリスト	1

フラグ	説明	適用される要素タイプ
11	インデント	1
12	アウトデント	1
13	左揃え	1
14	中央揃え	1
15	右揃え	1
16	イメージの挿入	1、5
17	リンクの作成	1、5
18	リンクの解除	1、5
19	<hr> の挿入	1
20	 の挿入	1
21	の挿入	1
22	静的リストへの新規行の追加	2
23	静的リストからの行の削除	2
24	静的リストの行の編集	2
25	静的リストの行を上へ移動	2
26	静的リストの行を下へ移動	2
27	フォントの種類の変更	1
28	フォント・サイズの変更	1
29	フォント色の変更	1
30	背景色の変更	1
31	（未使用）	
32	（未使用）	
33	（未使用）	
34	（未使用）	
35	（未使用）	
36	動的リストの次のページへ移動	3
37	動的リストの前のページへ移動	3
38	動的リストの表示列の変更	3
39	（未使用）	
40	動的リストのリフレッシュ	3
41	動的リストの新規コントリビュータ・データ・ファイルの作成	3
42	動的リストからの除外	3
43	動的リストで選択したアイテムのドキュメント情報の表示	3
44	スペル・チェッカ	1、6
45	必須	1、4、5、6
46	選択テキストへの CSS クラス名の適用	1、2、5

フラグ	説明	適用される要素タイプ
47	ハイパーリンクで指定されているか、動的リストに含まれているコントリビュータ・データ・ファイル（またはネイティブ・ドキュメント）の編集	1、3、5
48	（未使用）	
49	動的リストへの登録	3
50	コントリビュータ表のサポートの有効化	1
51	プレーン・テキストのみを貼り付けて書式を削除	1
52	オブジェクト・プロパティ・フォームの有効化	1、5、6
53	要素検証の有効化	1、2、5、6
54	アクセシビリティ・レポートの有効化	1、2、3、4、5、6
55	ソース・モードの有効化	1
56	全画面編集の有効化	1
57	動的リスト切替えの有効化	3
58	HTML タグの有効化	1
59	高さ設定の有効化	1、4、5

<element> タグには、フラグメント・パラメータが格納されたサーバー側変数を宣言する Idoc スクリプト・タグ（HTML コメント構文内に配置）が含まれます。このスクリプト・タグに含まれる正確な Idoc スクリプト・コードは、要素のタイプに応じて異なります。

4.6.1 要素タイプ : WYSIWYG

WYSIWYG 要素（type=1）では、宣言された Idoc 変数に、この要素のコンテンツをコントリビュータ・データ・ファイルから抽出する際に使用される XPath 式が含まれます。変数名は、要素識別子と同じです。値には、XPath 式を作成するためのコントリビューション・リージョンおよび要素の名前が使用されます。

たとえば、コントリビューション・リージョンの名前が story で、要素の名前が author の場合、Idoc 変数は次のようになります。

```
<!--$
   region1_element1= "story/author"
-->
```

4.6.2 要素タイプ : 静的リスト

静的リスト要素（type=2）では、Idoc 変数に単純な要素タイプと同じ XPath 式が含まれます。ただし、この XPath 式はコントリビュータ・データ・ファイル内の静的リストのルート・タグを示します（実際のコンテンツは、このルートの子要素内に含まれます）。

この Idoc 変数に加え、静的リスト・フラグメント・パラメータが、要素識別子とパラメータ名を使用して Idoc 変数として宣言されます。最終的には、ssIncludeXml() メソッド・コールを使用して、静的リスト・フラグメントを定義するフラグメント定義ファイルから <ssinfo> XML データ・アイランドに静的リストの子要素を取得する必要があります。

XML フラグメント定義ファイルに対するすべての ssIncludeXml() コールと同様に、フラグメントでフラグメント・パラメータ値を動的に取得するため、コールの直前に特別な Idoc 変数の ssFragmentInstanceId を定義する必要があります。

たとえば、リージョン名が story、要素名が relatedlinks、静的リスト識別子が listoflinks、およびその定義を含むフラグメント・ライブラリが myfragments の場合、Idoc 変数は次のようになります。

```
<!--$
    region1_element2 = "story/relatedlinks",
    region1_element2_param1 = "value1",
    region1_element2_param2 = "value2",
     . . .
    ssFragmentInstanceId = "region1_element2",
    ssIncludeXml("myfragments",
       "fragments/fragment[@id='listoflinks']/elements/node()")
-->
```

4.6.3 要素タイプ : 動的リスト

動的リスト要素（type=3）では、コントリビュータ・データ・ファイルに実際のコンテンツは格納されないため、Idoc 変数に他のタイプと同じ XPath 式は含まれません。かわりに、Idoc 変数には、動的リスト・フラグメント・パラメータが含まれます（要素識別子とパラメータ名が使用されます）。

たとえば、Idoc 変数は次のようになります。

```
<!--$
    region1_element3_param1 = "value1",
    region1_element3_param2 = "value2",
     . . .
-->
```

4.6.4 要素タイプ : カスタム

カスタム要素（type=4）では、宣言された Idoc 変数に、この要素のコンテンツをコントリビュータ・データ・ファイルから抽出する際に使用される XPath 式が含まれます。変数名は、要素識別子と同じです。値には、XPath 式を作成するためのコントリビューション・リージョンおよび要素の名前が使用されます。

たとえば、コントリビューション・リージョンの名前が story で、カスタム要素の名前が content の場合、Idoc 変数は次のようになります。

```
<!--$
   region1_element1= "story/content"
-->
```

4.6.5 要素タイプ : イメージ

イメージ要素（type=5）では、宣言された Idoc 変数に、この要素のコンテンツをコントリビュータ・データ・ファイルから抽出する際に使用される XPath 式が含まれます。変数名は、要素識別子と同じです。値には、XPath 式を作成するためのコントリビューション・リージョンおよび要素の名前が使用されます。

たとえば、コントリビューション・リージョンの名前が story で、要素の名前が photo の場合、Idoc 変数は次のようになります。

```
<!--$
   region1_element1= "story/photo"
-->
```

4.6.6 要素タイプ : プレーン・テキスト

プレーン・テキスト要素（type=6）では、宣言された Idoc 変数に、この要素のコンテンツをコントリビュータ・データ・ファイルから抽出する際に使用される XPath 式が含まれます。変数名は、要素識別子と同じです。値には、XPath 式を作成するためのコントリビューション・リージョンおよび要素の名前が使用されます。

たとえば、コントリビューション・リージョンの名前が story で、要素の名前が copyright の場合、Idoc 変数は次のようになります。

```
<!--$
   region1_element1= "story/copyright"
-->
```

4.7 <element> タグの子

<element> タグは、次の子タグを格納できます。

- 4-10 ページの「<description>」
- 4-11 ページの「<linktoregioncontent>」
- 4-12 ページの「<dynlistaddregioncontent>」
- 4-12 ページの「<insertimagequerytext>」
- 4-13 ページの「<querytext>」
- 4-13 ページの「<sortfield>」
- 4-13 ページの「<sortorder>」
- 4-14 ページの「<limitscope>」
- 4-14 ページの「<resultcount>」
- 4-14 ページの「<targetnodeid>」
- 4-14 ページの「<url>」
- 4-15 ページの「<classes>」
- 4-15 ページの「<validate>」

4.7.1 <description>

<description> タグには、個々のコントリビュータが Site Studio コントリビュータで要素名の上にポインタを置いたときに表示される、実体エンコードされた要素の説明が含まれます（たとえば、< のかわりに < が使用されます）。このタグは属性を持ちません。

4.7.2 <linktoregioncontent>

<linktoregioncontent> タグは、管理対象データ・ファイルまたはネイティブ・ドキュメントのハイパーリンクを作成する際に、コントリビュータが実行できるアクションを表します。このタグの属性は次のとおりです。

- **createnewxml:** true または false に設定し、リンクのターゲットとなる新規データ・ファイルの作成をコントリビュータに許可するかどうかを指定します。
- **createnewnative:** true または false に設定し、リンクのターゲットとなる新規ネイティブ・ドキュメントの作成をコントリビュータに許可するかどうかを指定します。
- **choosemanaged:** true または false に設定し、リンクのターゲットとなる既存の管理対象アイテムの選択をコントリビュータに許可するかどうかを指定します。
- **chooselocal:** true または false に設定し、新規管理対象アイテムとしてチェックインされ、リンクのターゲットとなる既存のローカル・ファイル・システム・ベースのアイテムの選択をコントリビュータに許可するかどうかを指定します。

子タグ

<linktoregioncontent> タグは、次の子タグを格納できます。

- **<createnewnativedoctypes>**

  <createnewnativedoctypes> タグは、ネイティブ・ドキュメントのハイパーリンクを作成する際に、コントリビュータが作成できるネイティブ・ドキュメントのタイプを表します。このタグは、属性を持ちませんが、ドキュメント拡張子のカンマ区切りリストを含む単一の CDATA 子セクションを格納する必要があります。たとえば、次のようになります。

  ```
  <createnewnativedoctypes>
      <![CDATA[.doc,.ppt,.txt,.xls]]>
  </createnewnativedoctypes>
  ```

- **<choosemanagedquerytext>**

  <choosemanagedquerytext> タグは、ハイパーリンクのターゲットとなる既存の管理対象アイテムの選択をコントリビュータに許可する場合に使用する問合せ文字列を表します。このタグの属性は次のとおりです。

  - **corecontentonly:** true の場合、検索結果はこの問合せ文字列のみに制限されます。false の場合、コントリビュータはカスタム検索を実行できます。

  <choosemanagedquerytext> タグは、問合せテキスト文字列を含む単一の CDATA 子セクションを格納する必要があります。たとえば、次のようになります。

  ```
  <choosemanagedquerytext>
      <![CDATA[dDocType <matches> 'ADACCT']]>
  </choosemanagedquerytext>
  ```

- **<defaultmetadata>**

  <defaultmetadata> タグは、ハイパーリンクのターゲットとなる新規管理対象アイテムの作成をコントリビュータに許可する場合に使用するデフォルト・メタデータを表します。このタグは、属性を持ちませんが、URL エンコード済のデフォルト・メタデータ文字列を含む単一の CDATA 子セクションを格納する必要があります。

  たとえば、次のようになります。

  ```
  <defaultmetadata>
      <![CDATA[dDocType%3DADACCT%26dDocType%3AisHidden%3Dtrue]]>
  </defaultmetadata>
  ```

<linktoregioncontent> タグは、<element> タグの非推奨の子タグ <linkaddmetadata> および <linkbrowsequerytext> にかわるものです。

これは、WYSIWYG、イメージおよび静的リスト・タイプの要素にのみ適用されます。

4.7.3 <dynlistaddregioncontent>

<dynlistaddregioncontent> タグは、動的リストの一部となる新規管理対象データ・ファイルまたはネイティブ・ドキュメントを作成する際に、コントリビュータが実行できるアクションを表します。このタグの属性は次のとおりです。

- **createnewxml:** true または false に設定し、動的リストの一部となる新規データ・ファイルの作成をコントリビュータに許可するかどうかを指定します。
- **createnewnative:** true または false に設定し、動的リストの一部となる新規ネイティブ・ドキュメントの作成をコントリビュータに許可するかどうかを指定します。
- **chooselocal:** true または false に設定し、新規管理対象アイテムとしてチェックインされ、動的リストの一部となる既存のローカル・ファイル・システム・ベースのアイテムの選択をコントリビュータに許可するかどうかを指定します。

子タグ

<dynlistaddregioncontent> タグは、次の子タグを格納できます。

- **<createnewnativedoctypes>**

  <createnewnativedoctypes> タグは、ネイティブ・ドキュメントのハイパーリンクを作成する際に、コントリビュータが作成できるネイティブ・ドキュメントのタイプを表します。このタグは、属性を持ちませんが、ドキュメント拡張子のカンマ区切りリストを含む単一の CDATA 子セクションを格納する必要があります。たとえば、次のようになります。

```
<createnewnativedoctypes>
    <![CDATA[.doc,.ppt,.txt,.xls]]>
</createnewnativedoctypes>
```

- **<defaultmetadata>**

  <defaultmetadata> タグは、ハイパーリンクのターゲットとなる新規管理対象アイテムの作成をコントリビュータに許可する場合に使用するデフォルト・メタデータを表します。このタグは、属性を持ちませんが、URL エンコード済のデフォルト・メタデータ文字列を含む単一の CDATA 子セクションを格納する必要があります。

  たとえば、次のようになります。

```
<defaultmetadata>
    <![CDATA[dDocType%3DADACCT%26dDocType%3AisHidden%3Dtrue]]>
</defaultmetadata>
```

<dynlistaddregioncontent> タグは、<element> タグの非推奨の子タグ <dynlistaddmetadata> にかわるものです。

これは、動的リスト・タイプの要素にのみ適用されます。

4.7.4 <insertimagequerytext>

<insertimagequerytext> タグには、コントリビュータのイメージ専用要素と、標準の WYSIWYG 要素のイメージ挿入機能で使用される事前定義された問合せテキスト文字列が含まれます。これは、フラグ 16 の設定により有効化されます（4-6 ページの「<element> タグ」のフラグの説明を参照）。このタグは、CDATA セクション内に問合せテキスト文字列を含み、オプションで次の属性を格納できます。

- **corecontentonly:** 表示される検索結果画面にコア・コンテンツのみを含める場合（つまり、検索結果のみを表示して Content Server の UI を表示しない場合）、TRUE に設定します。

これは、WYSIWYG、イメージおよび静的リスト・タイプの要素にのみ適用されます。

4.7.5 <querytext>

<querytext> タグは、動的リスト要素（type=3）専用であり、動的リストの検索基準が含まれます。このタグは、CDATA セクション内に検索基準を格納します。このタグは属性を持ちません。

Site Studio は、フラグメント・パラメータとして Idoc 変数に定義されている動的リストの検索基準を読み取って、このタグにデータを移入します。そのため、このタグには、Idoc 変数への参照のみが含まれます。ただし、問合せテキスト文字列内に埋め込まれた Idoc が、クライアント側のコントリビュータ・アプリケーションに渡される前に正しく評価されるように、参照は Idoc の eval() 文の中に格納されます。

たとえば、次のようになります。

```
<querytext>
    <![CDATA[<!--$eval(region1_element1_ssQueryText)-->]]>
</querytext>
```

これは、動的リスト・タイプの要素にのみ適用されます。

4.7.6 <sortfield>

<sortfield> タグは、動的リスト要素（type=3）専用であり、動的リストの検索基準のソート・フィールド名が含まれます。このタグは、CDATA セクション内にフィールド名を格納します。このタグは属性を持ちません。

Site Studio は、フラグメント・パラメータとして Idoc 変数に定義されている動的リストのソート・フィールドを読み取って、このタグにデータを移入します。そのため、このタグには、Idoc 変数への参照のみが含まれます。

たとえば、次のようになります。

```
<sortfield>
    <![CDATA[<!--$region1_element1_ssSortField-->]]>
</sortfield>
```

これは、動的リスト・タイプの要素にのみ適用されます。

4.7.7 <sortorder>

<sortorder> タグは、動的リスト要素（type=3）専用であり、動的リストの検索基準のソート順序が含まれます。このタグは、CDATA セクション内にソート順序の値を格納します。このタグは属性を持ちません。

Site Studio は、フラグメント・パラメータとして Idoc 変数に定義されている動的リストのソート順序を読み取って、このタグにデータを移入します。そのため、このタグには、Idoc 変数への参照のみが含まれます。

たとえば、次のようになります。

```
<sortorder>
    <![CDATA[<!--$region1_element1_ssSortOrder-->]]>
</sortorder>
```

これは、動的リスト・タイプの要素にのみ適用されます。

4.7.8 <limitscope>

<limitscope> タグは、動的リスト要素（type=3）専用です。このタグには、動的リストの結果として自動的に描画するコンテンツを、コンテンツ・サーバーのすべてから取得するか、現在のサイト階層のみから取得するかを指定する true または false の値が含まれます。このタグは、CDATA セクション内に true または false の値を格納します。このタグは属性を持ちません。

Site Studio は、フラグメント・パラメータとして Idoc 変数に定義されている動的リストの limitscope 値を読み取って、このタグにデータを移入します。そのため、このタグには、Idoc 変数への参照のみが含まれます。

たとえば、次のようになります。

```
<limitscope>
    <![CDATA[<!--$region1_element1_ssLimitScope-->]]>
</limitscope>
```

これは、動的リスト・タイプの要素にのみ適用されます。

4.7.9 <resultcount>

<resultcount> タグは、動的リスト要素（type=3）専用であり、動的リストの結果件数が含まれます。このタグは、CDATA セクション内に結果件数の値を格納します。このタグは属性を持ちません。

Site Studio は、フラグメント・パラメータとして Idoc 変数に定義されている動的リストの結果件数を読み取って、このタグにデータを移入します。そのため、このタグには、Idoc 変数への参照のみが含まれます。

たとえば、次のようになります。

```
<resultcount>
    <![CDATA[<!--$region1_element1_ssResultCount-->]]>
</resultcount>
```

これは、動的リスト・タイプの要素にのみ適用されます。

4.7.10 <targetnodeid>

<targetnodeid> タグは、動的リスト要素（type=3）専用であり、動的リストに使用されるターゲット・ノードが含まれます。このタグは、CDATA セクション内にターゲット・ノード ID の値を格納します。このタグは属性を持ちません。

Site Studio は、フラグメント・パラメータとして Idoc 変数に定義されている動的リストのターゲット・ノード ID を読み取って、このタグにデータを移入します。そのため、このタグには、Idoc 変数への参照のみが含まれます。

たとえば、次のようになります。

```
<targetnodeid>
    <![CDATA[<!--$region1_element1_ssTargetNodeId-->]]>
</targetnodeid>
```

これは、動的リスト・タイプの要素にのみ適用されます。

4.7.11 <url>

<url> タグには、コントリビュータがコンテンツを追加できるように GUI および検証ロジックを提供する管理対象 HCSP ページに対する相対 URL が含まれます。このタグは、CDATA セクション内に URL の値を格納します。このタグは属性を持ちません。

たとえば、次のようになります。

```
<url><![CDATA[groups/public/documents/adacct/myform.hcsp]]></url>
```

これは、カスタム・タイプの要素にのみ適用されます。

4.7.12 <classes>

<classes> タグには、コントリビュータがテキストの選択に適用できる CSS クラス名を格納した 1 つ以上の <class> タグが含まれます。これらのタグは属性を持ちません。

たとえば、次のようになります。

```
<classes>
   <class>code</class>
   <class>body</class>
   <class>footer</class>
</class>
```

これは、WYSIWYG、イメージおよび静的リスト・タイプの要素にのみ適用されます。

4.7.13 <validate>

<validate> タグは、オプションであり、要素をコントリビュータ・データ・ファイルに挿入する前に検証するためのクライアント側スクリプトの外部ファイルを表します。スクリプト・ルーチンは、任意の WSH 互換スクリプト言語（VBScript や JScript）で記述できます。このタグは、検証メソッドとして使用する関数を格納する必要があります。この関数では、入力として、検証対象の要素のコンテンツを含む 1 つ以上のパラメータを使用します。

戻り値

関数の戻り値は、次のいずれかになります。

- **ブール :** 要素コンテンツが有効か無効かを表す true または false の値。

  または

- **文字列 :** 要素コンテンツが有効の場合、空の文字列を戻し、要素コンテンツが無効の場合、エラー・メッセージを戻します。

属性

<validate> タグには、次の属性が含まれます。

- **docname:** 検証ロジックを含む管理対象スクリプト・ファイルの dDocName。
- **language:** 管理対象スクリプト・ファイルの構文。次のいずれかの値である必要があります。
  - javascript
  - vbscript
- **methodname:** この要素コンテンツを検証するためにコールするメソッドの名前。
- **paramtype:** 最初のパラメータとしてメソッドに渡される予定の要素コンテンツの形式。次のいずれかの値である必要があります。
  - **string:** 要素コンテンツは、メソッドの最初のパラメータとして、要素の HTML RAW テキストを含んだ文字列形式で渡されます。
  - **html:** 要素コンテンツは、メソッドの最初のパラメータとして、最も外側に HTMLElement を含んだ HTML DOM オブジェクト形式で渡されます。
  - **xml:** 要素コンテンツは、メソッドの最初のパラメータとして、最も外側に XMLDOMNode を含んだ XML DOM オブジェクト形式で渡されます。

子タグ

<validate> タグは、次の子タグを格納できます。

- **<parameters>**

  <parameters> タグは、常になんらかの形式の要素コンテンツを含む最初のパラメータの他に検証メソッドに渡される追加パラメータのコレクションを表します。これにより、単一の検証メソッドをパラメータ化して、異なる結果を生成する複数の要素で使用できます。

  <parameters> タグは、属性を持ちませんが、0（ゼロ）個以上の <parameter> タグを格納します。

  <parameter> タグは、検証メソッドに渡される個々の追加パラメータの値を表します。<parameter> タグは、属性を持ちませんが、そのコンテンツとしてパラメータ値を格納する必要があります。

  たとえば、次のようになります。

```
<parameters>
   <parameter>50</parameter>
   <parameter>January</parameter>
   <parameter>***-***-***</parameter>
</parameters>
```

4.8 <fragmentinstance> タグ

<fragmentinstance> タグは、レイアウト・ページのフラグメントの単一のインスタンスを表します。このタグの属性は次のとおりです。

- **id:** レイアウト・ページのフラグメントのこのインスタンスに対応する一意の識別子
- **fragmentid:** フラグメント実装の一意の識別子
- **library:** フラグメント実装を含むフラグメント・ライブラリの一意の識別子

<fragmentinstance> タグには、フラグメント・インスタンス識別子とパラメータ名を使用してフラグメント・パラメータ値を保持するサーバー側変数を宣言する Idoc スクリプト・タグ（HTML コメント・タグ内に配置）が含まれます。

たとえば、次のようになります。

```
<!--$
    fragment1_param1 = "value1",
    fragment1_param2 = "value2",
     . . .
-->
```

4.9 デザイナ用の追加のフラグメント・マークアップ

Site Studio では、コントリビューション・リージョンの構造を提供する複数の特殊なマーカーを、レイアウト・ページの <ssinfo> XML データ・アイランドに格納します。レイアウト・ページのマークアップ以外にも、フラグメント・スニペット、コントリビューション・リージョンおよび要素のすべてには、Site Studio デザイナで管理できるように最初と最後に特殊なマークアップが含まれます。

- 4-17 ページの「フラグメント・スニペットのマーカー」
- 4-18 ページの「リージョンのマーカー」
- 4-18 ページの「要素のマーカー」

4.9.1 フラグメント・スニペットのマーカー

レイアウト・ページの <ssinfo> XML データ・アイランドでは、フラグメント・スニペットを <fragmentinstance> タグで識別できます。このマークアップに加え、デザイナで管理できるようにフラグメント・スニペット自体の最初と最後もマークされます。(唯一の例外は、単純なスニペットです。このスニペットは、レイアウト・ページに配置されるとフラグメント・スニペットとして扱われなくなります。)

たとえば、次のようになります。

```
<!-- SS_BEGIN_SNIPPET(fragmentid, snippetid)-->
     . . . snippet code goes here . . .
<!-- SS_END_SNIPPET(fragmentid, snippetid)-->
```

スニペットのインクルード属性が inline の場合、スニペット・コンテンツはこれらのマーカー間にインライン配置されます。スニペットのインクルード属性が reference の場合、スニペット・コンテンツに対する参照がこれらのマーカー間にインライン配置されます。この処理は、ssIncludeXml() スクリプト拡張機能を使用して行います。同時に、スニペット内のスクリプトにフラグメント識別子を提供するためにそのコールの直前で Idoc 変数の ssFragmentInstanceId を宣言します。

たとえば、フラグメント・インスタンス ID が fragment1、フラグメント ID が myfragment、スニペット ID が 1、およびフラグメント実装を格納するライブラリが myfragments の場合、コードは次のようになります。

```
<!-- SS_BEGIN_SNIPPET(fragment1, 1)-->
<!--$
  ssFragmentInstanceId="fragment1",
  ssIncludeXml("myfragments",
  "fragments/fragment[@id='myfragment']/snippets/snippet[@id='1']/text()")
-->
<!-- SS_END_SNIPPET(fragment1, 1)-->
```

4.9.2 リージョンのマーカー

レイアウト・ページの <ssinfo> XML データ・アイランドでは、コントリビューション・リージョンを <region> タグで識別できます。このマークアップに加え、デザイナで管理できるようにリージョン自体の最初と最後もマークされます。

次の 2 つのマーカーがあります。

- 開始リージョン・マーカー
- 終了リージョン・マーカー

開始リージョン・マーカー

リージョンの開始は、重要な変数の Idoc 宣言と、ss_open_region_definition リソース・インクルードのインクルードを含む開始リージョン・マーカーでマークされます。

たとえば、次のようになります。

```
<!-- SS_BEGIN_OPENREGIONMARKER(region1)-->
  <!--$SS_REGIONID="region1" -->
  <!--$include ss_open_region_definition -->
<!-- SS_END_OPENREGIONMARKER(region1)-->
```

Idoc 変数では、ss_open_region_definition リソース・インクルード内で使用される標準の SS_REGIONID 変数としてリージョン識別子を宣言します。

終了リージョン・マーカー

リージョンの終了は、終了リージョン・マーカーでマークされます。このマーカーには、ss_close_region_definition リソース・インクルードのインクルードが含まれます。

たとえば、次のようになります。

```
<!-- SS_BEGIN_CLOSEREGIONMARKER(region1)-->
  <!--$include ss_close_region_definition -->
<!-- SS_END_CLOSEREGIONMARKER(region1)-->
```

4.9.3 要素のマーカー

レイアウト・ページの <ssinfo> XML データ・アイランドでは、3 つの単純な要素（WYSIWYG、プレーン・テキストおよびイメージ）を <element> タグで識別できます。このマークアップに加え、デザイナで管理できるように要素自体の最初と最後もマークされます。マーカーには、コントリビュータ・データ・ファイルから実際のコントリビューション・コンテンツを取得してレイアウト・ページに表示するため、ssIncludeXml() メソッドのコールが含まれます。

たとえば、要素 ID が region1_element1 の場合（この名前は、XPath 式を含む <ssinfo> XML データ・アイランドで Idoc 変数を宣言するためにも使用されます）、コードは次のようになります。

```
<!--SS_BEGIN_ELEMENT(region1_element1)-->
   <!--$ssIncludeXml(SS_DATAFILE, region1_element1 & "/node()")-->
<!--SS_END_ELEMENT(region1_element1)-->
```

2 つの拡張要素（静的リストおよび動的リスト）の場合、どちらの要素タイプもフラグメントとして実装されるため、レイアウト・ページに配置されるマークアップは、フラグメント・インスタンス・マーカーと同じです。

5
フラグメント

この章の内容は次のとおりです。

5.1 フラグメントについて

フラグメントは、HTML またはスクリプト（クライアント側の JavaScript、およびサーバー側の Idoc、ASP または JSP を含む）の自己完結型のスニペットであり、Web サイトの複数のレイアウト・ページで再利用可能な値を含むことができます。

フラグメントは、<IMG SRC=xxx>、<SCRIPT SRC=xxx>、<$ docLoadResource(xxx) $> などのタグを使用して他のファイルへの参照も格納できます。外部参照されるファイルやリソースは、フラグメント・アセットとして機能するため、フラグメントの使用時には常に参照可能である必要があります。

単純なフラグメントは、レイアウト・ページの任意の場所に挿入できるアトミック・コンテンツを格納できます。より複雑なフラグメントでは、一部のコンテンツをページの <head> に配置し、他のコンテンツを <body> に配置する必要があります。<body> では、コンテンツを先頭や末尾に、またはカーソルの現在の位置に配置できます。つまり、フラグメントには、複数のフラグメント・スニペットを格納できます。

設計者は、Web サイトにおけるフラグメントの使用場所に応じて、フラグメントのルック・アンド・フィールまたは動作を変更できます。この機能は、フラグメント・パラメータの使用により可能になります。フラグメント・パラメータにより、フラグメントの作成者は特定の変数パラメータを指定できます。サイト設計者は、フラグメントをレイアウト・ページに実際に追加するときに、これらのパラメータを選択できます。

大きく分けると、フラグメントには次のものが含まれます。

- 0（ゼロ）個以上のフラグメント・パラメータ定義（名前、タイプ、デフォルト値）。これらは、フラグメント・スニペットで参照されますが、フラグメント・インスタンスを含むページごとに一意に宣言されます。
- 1 つ以上のフラグメント・スニペット。これらは、フラグメントをレイアウト・ページに追加する際にコンテンツを配置する必要のある場所を示す識別子を備えた、HTML またはスクリプトの断片です。
- 0（ゼロ）個以上のフラグメント・アセット。これらは、Web ページでの使用時にフラグメントから参照できるローカル・ファイルです。

フラグメントは、一般的に 2 つのスニペットで構成されます。1 つ目のスニペットは、通常、ページの <head> に存在し、ページを書式設定する CSS ファイルを参照するか、フラグメントの実装の一部または全部を提供する JavaScript ファイルを参照します。2 つ目のスニペットは、通常、ページの <body> のドロップ・ポイントに存在し、フラグメントのプレゼンテーションを含みます。このスニペットは、インクルードされた .js ファイルにより提供されるメソッドの JavaScript コールのように単純なこともあれば、HTML、JavaScript および Idoc スクリプトのコレクションを格納することもあります。

5.2 フラグメント・ライブラリ

Site Studio では、個々のフラグメントをフラグメント・ライブラリに格納します。フラグメントごとに独自のライブラリに格納することや、関連するフラグメントをまとめて同じライブラリに格納することが可能です。フラグメント・ライブラリは、次のような管理対象オブジェクトとしてコンテンツ・サーバーに格納されます。

- **プライマリ・ファイル :** フラグメント・ライブラリの各フラグメントにより必要とされるすべてのアセットを格納した ZIP ファイル
- **代替ファイル :** フラグメント・ライブラリの各フラグメントの構造全体を定義したフラグメント定義ファイル

Content Server では、プライマリ・ファイルまたは代替ファイル（あるいはその両方）としてコンテンツ・アイテムを管理します（Content Server のヘルプを参照）。これらのファイルは、Site Studio のプライマリおよびセカンダリ・ページとは無関係です。混同しないように注意してください。

注意： 前述のプライマリ・ファイルとしてチェックインされるフラグメント・アセットの ZIP ファイルを、フラグメント・アセットとフラグメント定義ファイルの両方を含むフラグメント・ライブラリの ZIP ファイルと混同しないでください。フラグメント・ライブラリの ZIP ファイルは、デザイナのフラグメント・ライブラリのアップロードおよびダウンロード・ユーティリティで使用するもので、フラグメント・ライブラリをより簡単に管理できます。デザイナのフラグメント管理ユーティリティを使用せずにコンテンツ・サーバーの管理対象フラグメント・ライブラリに直接アクセスする場合は、フラグメント・アセットの ZIP ファイルを管理するだけで済みます。

フラグメント定義ファイルには、ライブラリ内の各フラグメントを定義する 1 つ以上の <fragment> 要素を格納した <fragments> ルート要素が含まれます。フラグメント定義ファイルの正確な構文は、「フラグメント定義ファイル」で説明します。詳細は、5-6 ページの「フラグメント定義ファイル」を参照してください。

コンテンツ・サーバーにフラグメント・ライブラリを追加する場合、コンテンツ・サーバーの標準のチェックイン・ページは使用しません。かわりに、デザイナによるフラグメント・ライブラリのアップロード機能を使用します。この機能により、管理対象コンテンツ・アイテムをチェックインし、フラグメント・アセットの ZIP ファイルのコピーを次の適切なランタイム weblayout ディレクトリに抽出します（<CS_name> はコンテンツ・サーバーの名前です）。

```
<CS_name>\weblayout\fragments
```

このパスは、次の 2 つの Idoc 変数のいずれかを使用して、フラグメントのサーバー側 Idoc スクリプトで参照できます。

- **HttpFragmentsRoot:** fragments フォルダの完全な HTTP パス
- **HttpRelativeFragmentsRoot:** fragments フォルダの相対 HTTP パス

5.3 読取り専用フラグメント・ライブラリ

製品に付属するフラグメント定義ファイル（サンプル・フラグメント）内の <fragments> ルート要素は、デザイナ・アプリケーション内で編集または削除されないように、読取り専用属性が設定されています。この属性を設定または解除する GUI は公開されていません。出荷時に設定済のフラグメントは、Site Studio の将来のリリースへのアップグレード時に上書きされる予定のため、変更されないように設計されています。

もちろん、設計者は、これらのフラグメントをコピーおよび編集して、独自のコピーに変更を加えることができます。

5.4 <ssinfo> XML データ・アイランドのフラグメント・インスタンスの構造

フラグメント・インスタンスとそのパラメータを定義する構造は、デザイナ・アプリケーションにより <ssinfo> XML データ・アイランドで管理されます。データ・アイランドには、レイアウト・ページのフラグメント・インスタンスごとに単一の <fragmentinstance> タグが含まれ、各タグにはフラグメント・インスタンス・パラメータを宣言する Idoc スクリプトが含まれます。

静的リストや動的リストなどの拡張フラグメント・タイプでは、<fragmentinstance> タグに追加の子タグを格納できます。これらのタグとその属性の正確な構文は、4-16 ページの「<fragmentinstance> タグ」を参照してください。

5.5 フラグメント・スニペットと ssIncludeXml()

各フラグメントには、フラグメント・ライブラリのフラグメント定義ファイルで定義された 1 つ以上のスニペットが含まれます。フラグメント・スニペットは、次の 3 つの方法でフラグメントにインクルードできます。

- **simple:** フラグメント・スニペットは、追加のマークアップなしにレイアウト・ページに直接追加され、フラグメントとして認識または管理されなくなります。
- **inline:** フラグメント・スニペットは、レイアウト・ページに直接追加されますが、フラグメントとして認識および管理されるように前後に特殊なマークアップが付きます。
- **reference:** フラグメント・スニペットは、レイアウト・ページに実際に追加されず、かわりにインクルード・ファイルのようにスニペットへの参照が追加されます（同時に、フラグメントとして認識および管理されるように前後にマークアップが付きます）。

ごく単純なスニペット以外には、インクルード・メカニズムとして reference を使用することを強くお薦めします。この方法であれば、フラグメントがサイト全体の多くのレイアウト内で使用される場合でも、スニペット・コンテンツを単一の場所で管理できます。インライン・スニペット・メカニズムは、JSP および ASP フラグメントを実装する際にスニペットをインクルードする方法として一般的に使用されます（第 10 章「JSP によるサーバー側スクリプト機能」および第 11 章「ASP によるサーバー側スクリプト機能」を参照）。

インラインおよび参照スニペットの前後に配置される特殊なマークアップの正確な構文は、4-17 ページの「フラグメント・スニペットのマーカー」を参照してください。参照によりインクルードされるスニペットは、ssIncludeXml() という新規スクリプト拡張機能を使用してレイアウト・ページに追加されます。

このスクリプト拡張機能により、管理対象 XML ファイルから要素をインクルードしてレイアウト・ページに配置する Idoc メカニズムが提供されます。ssIncludeXml() のパラメータには、フラグメント・ライブラリのフラグメント定義ファイルの dDocName と、抽出される XML ノードの XPath 式が含まれます。（その他のパラメータの詳細は、7-2 ページの「ssIncludeXml()」を参照してください。）抽出される XML ノードのコンテンツは、現在のレイアウト・ページの有効範囲で詳細に評価され、必要に応じて追加のサーバー側 Idoc スクリプトがインクルードされます。

5.6 フラグメント・インスタンスのパラメータ

フラグメントのインスタンスをレイアウト・ページに配置する場合、そのフラグメントに定義された各パラメータに値（空白も可）を指定する必要があります。この値は、<ssinfo> XML データ・アイランドの <fragmentinstance> タグのサーバー側 Idoc 変数に格納されます。フラグメント・インスタンスのパラメータを指定するための正確な構文は、4-16 ページの「<fragmentinstance> タグ」を参照してください。

フラグメントのパラメータのネーミング規則は、同じレイアウト・ページで同じフラグメントの複数のインスタンスを使用するために重要です。フラグメント・インスタンスのすべてのパラメータ変数には、先頭にフラグメント・インスタンス識別子が追加されます。たとえば、<fragmentinstance> の ID 属性が fragment1 で、フラグメントで ssColor というパラメータを定義する場合、この特定のフラグメント・インスタンスに対応するパラメータの値は、Idoc 変数の fragment1_ssColor として宣言されます。

フラグメント・インスタンスのパラメータ値を含む Idoc 変数の名前がフラグメント・インスタンス識別子に依存する場合、フラグメント・スニペットの実装でその値を参照するための方法が必要になります。これには、ssFragmentInstanceId という標準の Idoc 変数を使用します。デザイナでは、この変数が、インラインまたは参照スタイルのフラグメント・スニペットが使用される直前に定義されます。

この変数には、フラグメント・インスタンス識別子のサーバー側表現が含まれます。この識別子をフラグメント・スニペットで使用することで、実際のインスタンス・パラメータ値を動的に取得できます。これを行うには、Idoc の eval() メソッドを使用します。たとえば、次のようになります。

```
<!--$ssClassName = eval("<$" & ssFragmentInstanceId & "_" & "ssClassName" & "$>")-->
<!--$ssTextColor = eval("<$" & ssFragmentInstanceId & "_" & "ssTextColor" & "$>")-->
<!--$ssHoverColor = eval("<$" & ssFragmentInstanceId & "_" & "ssHoverColor" & "$>")-->
```

5.7 カスタム・セクション・プロパティを使用するフラグメント

設計者は、カスタム・セクション・プロパティを定義して、Web サイトのセクションごとに各プロパティに一意の値を割り当てることができます（『Oracle Universal Content Management Site Studio デザイナ・ガイド』を参照）。定義と値は、Web サイト・プロジェクト・ファイルに格納されます。（3-2 ページの「Web サイト・プロジェクト・ファイルについて」を参照してください。）

カスタム・セクション・プロパティは、それ自体では役に立ちません。カスタム・プロパティが役に立つのは、レイアウト・ページ内（より一般的には、フラグメント・スニペット内）でクライアント側またはサーバー側スクリプトに参照される場合のみです。カスタム・セクション・プロパティの値にアクセスする場合、クライアント側 JavaScript とサーバー側 Idoc スクリプトを使用する 2 つの主な方法があります。

5.7.1 クライアント側 JavaScript

クライアント側の実行時生成ファイル sitenavigation.js には、現在の Web サイト階層の定義を格納した NavNode オブジェクトの配列が含まれます（2-15 ページの「sitenavigation.js」を参照）。現在のセクションの値を含む各カスタム・セクション・プロパティは、cp_ で始まる名前を持つメンバー変数のそのセクションに対応する NavNode オブジェクト内に格納されます。

カスタム・セクション・プロパティのこれらのクライアント側表現が最もよく使用されるのは、ナビゲーション・フラグメント内です。ナビゲーション・フラグメントは、NavNode オブジェクトを反復処理し、cp_XXX メンバー変数の存在を検出して使用することで、表示されるナビゲーション・スキームをカスタマイズできます。cp_XXX メンバー変数は、存在する場合としない場合があるため、パラメータへのアクセスは、Site Studio で提供される次の 2 つの JavaScript メソッドを使用することでより簡単になります。

- **customSectionPropertyExists (prop):** カスタム・セクション・プロパティが存在するかどうかを示す true または false の値を戻します。このメソッドが true を戻した場合、カスタム・セクション・プロパティを直接使用できます。
- **getCustomSectionProperty (prop):** カスタム・セクション・プロパティの値が存在する場合、その値を戻します。存在しない場合、空の文字列を戻します。

これら 2 つのメソッドのアクションは、Site Studio に付属する CSP Sample Navigation（client）フラグメントで確認できます（『Oracle Universal Content Management Site Studio デザイナ・ガイド』を参照）。

5.7.2 サーバー側 Idoc スクリプト

サーバー側 Idoc スクリプトを使用してカスタム・セクション・プロパティにアクセスする場合、次の複数の方法を使用できます。

- **ssGetNodeProperty (name):** このスクリプト拡張機能は、現在の Web サイト・セクションの名前付きプロパティの値を取得します。
- **ssGetNodeProperty (nodeId, name):** このスクリプト拡張機能は、指定された Web サイト・セクションの名前付きプロパティの値を取得します。
- **SS_GET_ALL_NODE_PROPERTIES サービス :** このサービスは、指定された Web サイト・セクションのすべてのカスタム・セクション・プロパティのリストを取得します。

これらのアクションは、Site Studio に付属する CSP Sample Navigation（server）、CSP Sample Dynamic List および Sample CSP Page Title フラグメントで確認できます（『Oracle Universal Content Management Site Studio デザイナ・ガイド』のサンプル・フラグメントに関する項を参照）。

追加のフラグメント・マークアップ情報は、「<ssinfo> XML データ・アイランド」の章を参照してください。

ssGetNodeProperty スクリプト拡張機能で取得できるセクション属性のリストは、3-5 ページの「<section> タグ」を参照してください。

カスタム・セクション・プロパティは、sitenavigation.xml ファイルのサイト階層の XML レンディションにも格納されます（2-17 ページの「sitenavigation.xml」を参照）。そのため、サーバー側スクリプトを使用してこの XML ファイルを解析するナビゲーション・フラグメントを作成する場合、各セクションのカスタム・セクション・プロパティに対するアクセス権も保持している必要があります。

プロパティの名前は、引用符で囲む必要があります。つまり、特定のノードの label プロパティを取得する場合であれば、次のように記述します。

```
<!--$label = ssGetNodeProperty(nodeId, "label")-->
```

5.8 フラグメント定義ファイル

フラグメント定義ファイルは、純粋な XML ファイルであり、ファイルの割当て先のコントリビューション・リージョンからその構造を継承したコンテンツを格納しています。コーディングの観点からは、フラグメント・ライブラリのフラグメント定義ファイルは次のように表現されます。

```
<fragments>
  <fragment>

    <parameters>
     <parameter> param definition goes here </parameter>
     <parameter> another param def goes here </parameter>
    </parameters>

    <snippets>
     <snippet> HTML snippet goes here </snippet>
     <snippet> another HTML snippet goes here </snippet>
    </snippets>

    <elements>
     <element> element definition goes here </element>
     <element> another element def goes here </element>
    </elements>

    </fragment>

</fragments>
```

フラグメント定義ファイルには、通常、次のタグが含まれます。

5.8.1 <fragments>

<fragments> タグは、フラグメント・ライブラリを表します。このタグの属性は次のとおりです。

- **id:** フラグメント・ライブラリの一意の名前または識別子。
- **name:** フラグメント・ライブラリの表示名。
- **readonly:** True に設定すると、ライブラリのフラグメントはデザイナ・アプリケーションで編集できなくなります。

<fragments> タグには、1 つ以上の <fragment> 子タグのコレクションが含まれます。

5.8.2 <fragment>

<fragment> タグは、単一のフラグメント定義を表します。このタグの属性は次のとおりです。

- **id:** フラグメントの一意の名前または識別子。スニペットを参照によりインクルードする場合に XPath 式で使用されます。
- **name:** フラグメントの表示名。
- **language:** フラグメント実装のためのサーバー側言語。使用可能な言語は、**idoc**、**asp** または **jsp** です。
- **type:** このフラグメントのタイプ。フラグメントが表示されるデザイナ・ツールボックスのカテゴリを決定するのに使用されます。また、フラグメントが静的リストと動的リストのどちらであるかも決定されます。使用可能なタイプは次のとおりです。
  - navigation
  - staticlist
  - dynamiclist
  - other
  - icon: デザイナ・ツールボックスでこのフラグメント用に使用するアイコンの名前。事前定義された次のフラグメント・アイコンのリストから選択します。

アイコン名	イメージ
region	
wysiwyg	
plaintext	abc
image1	
image2	
list1	
list2	
list3	
list4	
list5	
list6	
tree	
horizontalrule	
nonbreakingspace	
linebreak	
span	‹sp›
div	‹dv›
heading1	‹h1›
heading2	‹h2›
heading3	‹h3›
heading4	‹h4›
heading5	‹h5›
heading6	‹h6›
copyright	
flash	
companylogo	
documents	

子タグ

<fragment> タグには、次の子タグが含まれます。

- **<parameters>:** 0（ゼロ）個以上のパラメータのコレクション。詳細は、5-9 ページの「<parameter>」を参照してください。
- **<snippets>:** 0（ゼロ）個以上の <snippet> のコレクション。詳細は、5-14 ページの「<snippet>」を参照してください。
- **<elements>:** 0（ゼロ）個以上の <element> のコレクション。詳細は、5-14 ページの「<element>」を参照してください。

5.8.3 <parameter>

<parameter> タグは、フラグメントの単一のパラメータを表します。このタグの属性は次のとおりです。

- **name:** パラメータ名。
- **type:** パラメータ・タイプ。次のものがあります。
  - **text:** 単純なテキスト型パラメータ。1 つ以上の <option> 子タグを使用して、事前定義済の選択項目のリストを格納できます。
  - **bigtext:** 複数行のテキスト型パラメータ。
  - **boolean:** 単純な True/False 型パラメータ。
  - **integer:** 単純な整数型パラメータ。
  - **float:** 単純な浮動小数点型パラメータ。
  - **size:** HTML サイズ（10px、50%、3pt など）を表すパラメータ。
  - **color:** 色の値（RGB または HTML の色名を使用）。
  - **url:** URL。
  - **manageddoc:** 「Search Results」ページで選択された、Content Server の管理対象ドキュメントの dDocName。問合せパラメータは、<querytext> 子タグを使用して指定できます。
  - **managedurl:** 「Search Results」ページで選択された、Content Server の管理対象ドキュメントの DocUrl。問合せパラメータは、<querytext> 子タグを使用して指定できます。
  - **managedquery:** 「CAPTURE_QUERY」ページで選択された問合せテキスト文字列。
  - **cssstyle:** CSS スタイル属性を表すテキスト・パラメータ。このタイプは、text タイプのように動作します。（将来的な使用を意図しています。）
  - **siteid:** 「Select Site」ダイアログ・ボックスで選択された、コンテンツ・サーバーの同じインスタンスに含まれるいずれかの Web サイトの siteId 値。
  - **nodeid:** 「Select Section」ダイアログ・ボックスのサイト階層を通じて選択された、Web サイトのいずれかのセクションの nodeId 値。
  - **custom:** <customgui> 子タグを使用して値を入力するための独自のカスタム GUI を提供するパラメータ。
- **description:** 「Fragment Parameter Values」ダイアログ・ボックスに表示されるパラメータの説明。
- **required:** レイアウト・ページにフラグメントを追加する場合にそのフラグメントのパラメータが必須であるかどうかを指定する true または false の値。
- **optionsonly:** 1 つ以上の <option> 子タグに使用可能な事前定義済のオプションが提供されている場合にのみ適用することを示す True または False の値。

<parameter> タグは、パラメータのデフォルト値（存在する場合）を表すテキストを格納できます。また、次のオプションの子タグも格納できます。

- **<option>:** text タイプのパラメータについて、1 つ以上のオプション・タグを使用して、事前定義済の選択項目のリストを指定します。詳細は、5-10 ページの「<option>」を参照してください。
- **<querytext>:** manageddoc および managedurl タイプのパラメータの QueryText 値が含まれます。詳細は、5-10 ページの「<querytext>」を参照してください。
- **<validate>:** パラメータを検証するためのカスタマイズされたスクリプト関数が含まれます。詳細は、5-11 ページの「<validate>」を参照してください。
- **<convert>:** パラメータを挿入する前に変換するためのカスタマイズされたスクリプト関数が含まれます。詳細は、5-11 ページの「<convert>」を参照してください。
- **<customgui>:** パラメータ値を入力するための独自の GUI を提供するカスタマイズされた HTML スニペットが含まれます。詳細は、5-12 ページの「<customgui>」を参照してください。

5.8.3.1 <option>

<option> タグは、選択リストの単一のオプションを表します。text タイプのパラメータにのみ適用されます。<option> タグは、選択リストに表示されるテキストを格納する必要があります。このタグの属性は次の 1 つです。

value: このオプションが選択されたときに挿入する値。この属性は、オプションに表示される値と挿入される値が同じ場合は任意指定です。

たとえば、次のようになります。

```
<option>Option 1</option>
<option>Option 2</option>
<option>Option 3</option>
<option>Option 4</option>
```

または

```
<option value="A">Option A</option>
<option value="B">Option B</option>
<option value="C">Option C</option>
<option value="D">Option D</option>
```

5.8.3.2 <querytext>

<querytext> タグは、コンテンツ・サーバーで検索を実行する場合に使用する問合せパラメータを表します。manageddoc または managedurl タイプのパラメータにのみ適用されます。このタグは、CDATA セクション内に問合せテキストを格納します。このタグは属性を持ちません。

たとえば、次のようになります。

```
<querytext>
  <![CDATA[
     dExtension <matches> `gif` <or> dExtension <matches> `jpg`
  ]]>
</querytext>
```

5.8.3.3 <validate>

<validate> タグは、オプションであり、パラメータをフラグメント・スニペットに挿入する前に検証するためのカスタマイズされたスクリプト・ルーチンを表します。スクリプト・ルーチンは、任意の WSH 互換スクリプト言語（VBScript や JScript）で記述できます。このルーチンには、単一の文字列を入力として使用する validate という単一の関数が含まれる必要があります。

関数の戻り値は、次のいずれかになります。

- **ブール :** パラメータが有効か無効かを表す True または False の値。

  または

- **文字列 :** パラメータが有効の場合、空の文字列を戻し、パラメータが無効の場合、エラー・メッセージを戻します。

<validate> タグは、CDATA セクション内にスクリプトを格納します。このタグの属性は次の 1 つです。

- **language:** このスクリプトで使用する言語（VBScript または JScript）

たとえば、次のようになります。

```
<validate language="VBScript">
  <![CDATA[
    Function validate(strInput)
     If InStr(strInput, "hello") > 0 Then
      validate = ""
     Else
      validate = "Error, ValidatedParam must contain " _
      "the text 'hello' somewhere in the string"
     End If
    End Function
  ]]>
</validate>
```

> **注意 :** <validate> タグは、使用できますが、「Fragment Editor」ユーザー・インタフェースには表示されません。つまり、現在はサポートされていません。

5.8.3.4 <convert>

<convert> タグは、オプションであり、パラメータをフラグメント・スニペットに挿入する際に変換するためのカスタマイズされたスクリプト・ルーチンを表します。スクリプト・ルーチンは、任意の WSH 互換スクリプト言語（VBScript や JScript）で記述できます。このルーチンには、単一の文字列を入力として使用し、変換後の文字列を出力として戻す convert という単一の関数が含まれる必要があります。

<convert> タグは、CDATA セクション内にスクリプトを格納します。このタグの属性は次の 1 つです。

- **language:** このスクリプトで使用する言語（VBScript または JScript）

たとえば、次のようになります。

```
<convert language="VBScript">
  <![CDATA[
    Function convert(strInput)
     If StrComp(strInput, "") = 0 Then
      convert = "(empty)"
     Else
      convert = "Your Value Was [" & strInput & "]"
     End If
    End Function
  ]]>
</convert>
```

注意： <convert> タグは、使用できますが、「Fragment Editor」ユーザー・インタフェースには表示されません。つまり、現在はサポートされていません。

5.8.3.5 <customgui>

<customgui> タグは、custom タイプのパラメータ専用です。このタグにより、フラグメント設計者は、フラグメント・パラメータを入力するための独自の（単純な）GUI を提供できます。このタグは、標準の HTML または JavaScript を含むことが可能な CDATA セクション内に HTML スニペットを格納します。

このタグは、4 つの JavaScript メソッドを通じて Site Studio と対話します。

- **window.external.GetValue():** パラメータの初期値を取得するメソッド。カスタム・パラメータは、プレーン・テキスト・パラメータと同じように処理を開始しますが、編集フィールドにはカスタム GUI を表示するためのボタン（図 5-1）があります。このメソッドは、編集フィールドから現在の値を取得します。

図 5-1 編集フィールドのカスタム GUI ボタン

- **window.external.SetValue():** 編集フィールドに新しい値を設定するメソッド。
- **window.external.OnOK():** Site Studio に対して、処理が終了し、SetValue() メソッドにより渡された値を承認する必要があることを指示するメソッド。
- **window.external.OnCancel():** Site Studio に対して、処理が終了したが、SetValue() メソッドにより渡された値を無視する必要があることを指示するメソッド（値が存在する場合）。

たとえば、次のようになります。

```
<customgui>
 <![CDATA[
  <HTML>
  <HEAD>
  <SCRIPT language="javascript">
   function initialize()
   {
    strResult = window.external.GetValue() + "00000";
    bold.checked = (strResult.charAt(0) == "1");
    italic.checked = (strResult.charAt(1) == "1");
    underlined.checked = (strResult.charAt(2) == "1");
    font.checked = (strResult.charAt(3) == "1");
   }
   function getvalue()
   {
    strResult = "";
    strResult = strResult + (bold.checked ? "1" : "0");
```

```
    strResult = strResult + (italic.checked ? "1" : "0");
    strResult = strResult + (underlined.checked ? "1" : "0");
    strResult = strResult + (font.checked ? "1" : "0");
    return strResult;
   }
  </SCRIPT>
  </HEAD>
  <BODY onload="initialize();">
    Welcome to my Customized Parameter Input Screen. This example
    could perhaps be used as a starting point for entering the
    contribution suppression flags.<BR>
    <TABLE>
     <TR>
      <TD>Allow BOLD</TD>
      <TD><INPUT type="checkbox" id="bold"></TD>
     </TR>
     <TR>
      <TD>Allow ITALIC</TD>
      <TD><INPUT type="checkbox" id="italic"></TD>
     </TR>
     <TR>
      <TD>Allow UNDERLINED</TD>
      <TD><INPUT type="checkbox" id="underlined"></TD>
     </TR>
     <TR>
      <TD>Allow Font Changes</TD>
      <TD><INPUT type="checkbox" id="font"></TD>
     </TR>
     <TR><TD COLSPAN="2"> </TD></TR>
     <TR>
      <TD COLSPAN="2">
       <INPUT type="button" value="OK"
         onclick="window.external.SetValue(getvalue()); window.external.OnOK();"/>
       <INPUT type="button" value="Cancel"
         onclick="window.external.OnCancel();"/>

      </TD>
     </TR>
    </TABLE>
   </BODY>
 ]]>
</customgui>
```

注意： <customgui> タグは、使用できますが、「Fragment Editor」ユーザー・インタフェースには表示されません。つまり、現在はサポートされていません。

5.8.4 <snippet>

<snippet> タグは、フラグメントの単一のスニペットを表します。このタグの属性は次のとおりです。

- **id:** フラグメント定義におけるスニペットの一意の識別子。
- **location:** スニペットを配置するレイアウト・ページの場所。次の 4 つの場所があります。
  - head
  - topofbody
  - drop-point（設計者がレイアウト・ページにフラグメントを配置する場所）
  - bottomofbody
- **include:** スニペットをレイアウト・ページにインクルードする方法。
  - **simple:** スニペット・コンテンツを直接レイアウト・ページにコピーし（フラグメント・インスタンスの詳細でスニペットをマークアップしません）、最初の挿入時にスニペット・コンテンツで %paramname% を検索してパラメータを直接インライン置換します。
  - **inline:** スニペット・コンテンツを直接レイアウト・ページにコピーし、フラグメント・スニペットとして単独で移動、編集および削除できるように、フラグメント・インスタンス・スニペットとしてコンテンツをマークします。
  - **reference:** ssIncludeXml() コールをレイアウト・ページに挿入し、フラグメント・スニペットとして単独で移動、編集および削除できるように、フラグメント・インスタンス・スニペットとしてそのコールをマークします。

<snippet> タグには、フラグメントの単一のアトミック・コンテンツを構成する HTML またはスクリプトの完全な単一のスニペットが含まれます。このスニペット・コンテンツは、CDATA セクション内に格納されます。また、このタグは、次のオプションの子タグを格納できます。

- **<designview: >** デザイナの「Design」ビューに表示されるスニペットの設計時表現

5.8.5 <designview>

<designview> タグは、デザイナの「Design」ビューに表示されるフラグメント・スニペットのオプションの設計時ビューを表します（デフォルト表示はフラグメント名です）。このタグは、CDATA セクション内に HTML を格納します。このタグは属性を持ちません。

たとえば、次のようになります。

```
<designview>
   <![CDATA[
     example copyright fragment goes here
   ]]>
</designview>
```

5.8.6 <element>

<element> タグは、静的リスト・フラグメント定義に格納された単純な要素を表します。<fragment> 定義タグにおける <element> タグの構文は、レイアウト・ページの <region> 定義タグで使用される <element> タグの構文と同じです。詳細は、4-6 ページの「<element> タグ」および 6-4 ページの「静的リスト」を参照してください。

静的リスト・フラグメント定義内の要素として使用できるのは、
タイプ 1、4、5、6（WYSIWYG、カスタム、イメージおよびプレーン・テキスト）の要素のみです。

6

コントリビューション・リージョンと要素

この章の内容は次のとおりです。

6.1 コントリビューション・リージョンと要素について

コントリビューション・リージョンと要素は、Site Studio の基本部分です。コントリビューション・リージョンは、特にコントリビュータによって提供されるコンテンツを格納するレイアウト・ページの領域です。コントリビューション・リージョンには、コントリビュータにより提供されるコンテンツの基本単位である 1 つ以上の要素が含まれます。（コントリビューション・リージョンと要素の詳細は、『Oracle Universal Content Management Site Studio デザイナ・ガイド』を参照してください。）

コントリビューション・リージョンに追加できる要素のタイプは、WYSIWYG、プレーン・テキスト、イメージおよびカスタムの 4 つです。カスタム要素は、他の要素と異なっており、完全にカスタマイズされたコントリビュータ環境を提供するために使用します。カスタム要素は、コントリビュータがコントリビュータ・アプリケーションに要素の値を入力できるように GUI と検証ロジックを提供する、外部の管理対象 HTML/HCSP ページ（Web ベース・フォーム）を参照するように設計されています。

また、フラグメントの形式で実装される次の 2 つのタイプの拡張要素があります。

- **静的リスト :** コントリビュータが使用して、3 つの単純な要素タイプ（WYSIWYG、プレーン・テキストおよびイメージ）の論理 2 次元配列を提供できます（コンシューマが参照することもできます）。詳細は、6-4 ページの「静的リスト」を参照してください。
- **動的リスト :** この要素は、コントリビュータが実際に情報を提供せず、かわりに設計者がコンテンツ・サーバーでコンテンツ・アイテムの検索基準を提供する特殊な場合に使用します。コントリビュータは、コンテンツ・アイテム（コントリビュータ・データ・ファイルとネイティブ・ドキュメント）をリストに追加する権限や、それらを削除する権限を持つことが可能です。詳細は、6-6 ページの「動的リスト」を参照してください。

6.2 コントリビュータ・モードでの Web ページの表示

Site Studio で作成された Web サイトには、最も新しくリリースされたバージョンのレイアウト・ページ、コントリビュータ・データ・ファイルおよびネイティブ・ドキュメントが表示され、サイト・ビジター（コンシューマ）は他の Web サイトと同じようにこれらのページを参照できます。コンテンツを Web サイトに追加（投稿）する場合、ユーザーは、Web ページを編集可能にする特殊なモードであるコントリビューション・モードに移行する必要があります。

Web ページを通常の表示からコントリビューション・モードに変更するには、キーボード・コマンド（ホット・キー）の組合せを使用します。デフォルトは、[Ctrl]+[Shift]+[F5] です。このホット・キー設定は、必要に応じて変更できます（詳細は『Oracle Universal Content Management Site Studio インストレーション・ガイド』を参照）。

ホット・キーにより、Cookie が書き込まれて Web ページが通常の表示からコントリビューション・モードに切り替わります。ホット・キーを再度入力すると、Cookie が削除されるため、ページはサイトのコンシューマに表示されるような通常の表示に戻ります。ブラウザがコントリビューション・モードに移行すると、Cookie 値の SSContributor=true の設定によりこのモードが維持されます。この方法により、ユーザーがリンクをクリックして現在のページから別のページにナビゲートしても、ブラウザではコントリビューション・モードが維持されます。

コントリビューション・モードで Web ページを表示すると、最新のワークフロー・バージョン（または将来のリリース・バージョン）のレイアウト・ページ、コントリビュータ・データ・ファイルおよびネイティブ・ドキュメントがユーザーに表示されます。ログイン・ユーザーに対するセキュリティ・チェックが実行され、ユーザーが編集できるコントリビュータ・データ・ファイルとネイティブ・ドキュメントが決定されます。各コントリビューション・リージョンの開始部分には、コントリビューション・アイコンが表示されます（そのリージョンが編集できる場合）。アイコンを右クリックすると、ユーザーのセキュリティに応じて、そのリージョンに関連する複数の選択項目が含まれたポップアップ・メニューが表示されます。

6.3 <ssinfo> XML データ・アイランドでのコントリビューション・リージョンの構造

コントリビューション・リージョンと要素を定義する構造は、デザイナ・アプリケーションにより <ssinfo> XML データ・アイランドで管理されます。データ・アイランドには、レイアウト・ページのコントリビューション・リージョンごとに単一の <region> タグが含まれ、各リージョンにはリージョンの要素ごとに単一の <element> タグが含まれます。これらのタグとその属性の正確な構文は、4-3 ページの「<region> タグ」および 4-6 ページの「<element> タグ」を参照してください。

6.4 レイアウト・ページでのコントリビューション・リージョンと要素のマークアップ

コントリビューション・リージョンとその要素の構造は、<ssinfo> XML データ・アイランドに格納されますが、各リージョンおよび（リージョン内の）要素の開始と終了は、レイアウト・ページの <body> でマークアップされます。このマークアップにより、レイアウト・ページでのコントリビューション・リージョンと要素の場所が示されます。リージョンと要素をマークアップするための正確な構文は、4-18 ページの「リージョンのマーカー」および 4-18 ページの「要素のマーカー」を参照してください。

特に重要な開始および終了リージョン・マーカーとして、次の 2 つのリソース・インクルードがあります。

- 6-3 ページの「ss_open_region_definition リソース・インクルード」
- 6-3 ページの「ss_close_region_definition リソース・インクルード」

6.4.1 ss_open_region_definition リソース・インクルード

ss_open_region_definition リソース・インクルードは、コントリビューション・リージョンの開始をマークするためにレイアウト・ページに挿入されます。このリソース・インクルードは、リージョンに割り当てられた管理対象コンテンツ・アイテム（コントリビュータ・データ・ファイルまたはネイティブ・ドキュメント）に対してセキュリティ・チェックを実行するためのサーバー側のロジックを格納しており、コントリビューション・リージョンに対して許可されるアクション（存在する場合）を決定し、コントリビュータ用の適切な右クリック・ポップアップ・メニューを備えたコントリビューション・アイコンを生成します。

セキュリティ・チェックに加え、ss_open_region_definition インクルードでは、ファイルがコントリビュータ・データ・ファイルとネイティブ・ドキュメントのどちらであるかがチェックされます。コントリビュータ・データ・ファイルである場合、コントリビューション・リージョンに挿入されます。ネイティブ・ドキュメントである場合、特殊なロジックが起動され、（Dynamic Converter を使用して）ドキュメントのインライン動的変換が実行されます。変換されたドキュメントにより、リージョンの構造全体が置き換えられ、要素は表示されなくなります。（これを行うため、通常のリージョン構造は HTML <comment> タグでラップされます。）

ss_open_region_definition リソース・インクルードの正確なロジックは、ss_resources.htm コンポーネント・リソース・ファイルで確認できます。このロジックでは、Site Studio コンポーネントで提供される Idoc スクリプト拡張機能が多く使用されます。これらの拡張機能の構文は、7-1 ページの「Idoc スクリプト拡張機能」を参照してください。

6.4.2 ss_close_region_definition リソース・インクルード

ss_close_region_definition リソース・インクルードは、各リージョンの終了をマークするためにレイアウト・ページにインクルードされます。このリソース・インクルードは、ss_open_region_definition リソース・インクルードのスクリプトと一致するサーバー側のロジックを格納しており、（たとえば、動的に変換されたネイティブ・ドキュメントでリージョンが置換される場合に）挿入された <comment> タグが正しく閉じられていることを保証します。

6.5 コントリビュータ・データ・ファイルと ssIncludeXml()

簡単にまとめると、コントリビューション・リージョンとその各要素の構造は、レイアウト・ページの <head> にある <ssinfo> XML データ・アイランドに定義されます。各リージョンの開始と終了を定義するマークアップは、レイアウト・ページの <body> 内にあります。リージョン境界内には、ハードコードされた HTML と複数の要素が混在することがあります。要素自体にはコンテンツは含まれません。要素は、コントリビュータ・データ・ファイルまたはデータ・ファイルと呼ばれる別個のファイルに維持されたコンテンツを参照します。

このファイルは、XML ファイルであり、ファイルの割当て先のコントリビューション・リージョンからその構造を継承したコンテンツを格納しています。コントリビュータ・データ・ファイルのコンテンツは、ssIncludeXml() スクリプト拡張機能を使用してレイアウト・ページに挿入されます。この新規拡張機能により、管理対象 XML ファイルからレイアウト・ページに HTML のスニペットとして要素をインクルードする Idoc メカニズムが提供されます。

ssIncludeXml() コールのパラメータには、XML ファイルの dDocName と、XML コンテンツをコントリビュータ・データ・ファイルから抽出するための XPath 式が含まれます。パラメータの詳細は、7-2 ページの「ssIncludeXml()」を参照してください。抽出されるコンテンツは、現在のレイアウト・ページの有効範囲で詳細に評価され、必要に応じて追加の Idoc スクリプトがインクルードされます。

注意： データ・ファイルの形式は、Site Studio 10.1.3.3.3 で変更されています。

6.6 拡張要素

Site Studio には、WYSIWYG、プレーン・テキスト、イメージ、カスタムなどの基本要素に加え、基本要素をフラグメントの機能と結合した拡張要素があります。拡張要素では、コントリビュータはコンテンツを提供しますが、そのコンテンツのプレゼンテーションは、フラグメントの形式で間接的に Web ページ上に配置されます。

次の 2 種類の拡張要素（フラグメント）があります。

- 静的リスト（6-4 ページの「静的リスト」を参照）
- 動的リスト（6-6 ページの「動的リスト」を参照）

6.6.1 静的リスト

静的リストは、コントリビュータ・データ・ファイルから抽出された反復コンテンツを表します。静的リストにより、設計者は、コントリビューション・コンテンツの論理 2 次元表を反復して提供できる 1 つ以上の単純な要素のセットを定義できます。静的リストのプレゼンテーションは、静的リスト・フラグメントで提供されます。

静的リスト・フラグメントは、各反復行のループを実装したサーバー側スクリプトを格納しており、ssIncludeXml() スクリプト拡張機能の使用により任意のスタイルで各要素を表示します。スクリプト拡張機能の XPath 式は、ループ・コードの反復回数に応じて動的に生成されます。

静的リストでは、コントリビューション・コンテンツの論理 2 次元表を使用できますが、コンテンツのプレゼンテーションに 2 次元表を使用する必要はありません。プレゼンテーションは、フラグメントを作成する個々のユーザーが決定できます。コンテンツは、カンマ区切り値、箇条書きリスト、表など、サーバー側スクリプトで生成できる任意の形式で表示できます。

プレゼンテーション・ロジックでループを実装するには、ssGetXmlNodeCount() スクリプト拡張機能を使用します。この新規スクリプト拡張機能は、反復回数をカウントし、ループのたびにコンテンツをインクルードするために ssIncludeXml() コールで使用される XPath 式を動的に生成します。

静的リストを実装するには、フラグメント定義に子要素を追加する必要があります。これを行うには、フラグメント定義ファイル内の <elements> コンテナ・タグに 1 つ以上の <element> タグを挿入します。これらの子要素の構文は、レイアウト・ページにおける通常の単純な要素の構文と同じです。

注意： 静的リストは、子要素が使用されるフラグメントの唯一のタイプです。

静的リストの例

製品情報の反復セットを、製品ごとの写真、名前、価格および説明付きで作成する場合、4 つの基本的な子要素を使用して静的リスト・フラグメントを設定します。次の図では、左側に写真があり、右側に製品名、価格および説明があります。これらはすべてプレゼンテーション全体の中で 1 つの行を構成します。

図 6-1 静的リストの例

Baseball Cap

$15.00

This low-profile, khaki baseball cap features the Stellent 'S' logo embroidered over the brim and the Stellent name across the back. One size fits most.

このプレゼンテーションを作成する場合、次のフラグメント・スニペットを使用できます。

```
<!--$root = eval("<$" & ssFragmentInstanceId & "$>")-->

<!--$strNumRecords=ssGetXmlNodeCount(SS_DATAFILE, root & "/name")-->
<!--$nNumRecords=toInteger(strTrimWs(strNumRecords))-->
<!--$nPos=1-->

<table border="0" width="100%" cellspacing="0" cellpadding="10">
 <tr><td> </td><td> </td></tr>

 <!--$loopwhile nPos <= nNumRecords -->

 <tr>
   <td rowspan="3">
     <!--$ssIncludeXml(SS_DATAFILE, root & "/photo[" & nPos & "]/node()")-->
   </td>
   <td>
     <!--$ssIncludeXml(SS_DATAFILE, root & "/name[" & nPos & "]/node()")-->
   </td>
 </tr>
 <tr>
   <td>
     <!--$ssIncludeXml(SS_DATAFILE, root & "/price[" & nPos & "]/node()")-->
   </td>
 </tr>
 <tr>
   <td>
     <!--$ssIncludeXml(SS_DATAFILE, root & "/desc [" & nPos & "]/node()")-->
   </td>
 </tr>

 <!--$ nPos = nPos + 1-->
 <!--$endloop-->

</table>
```

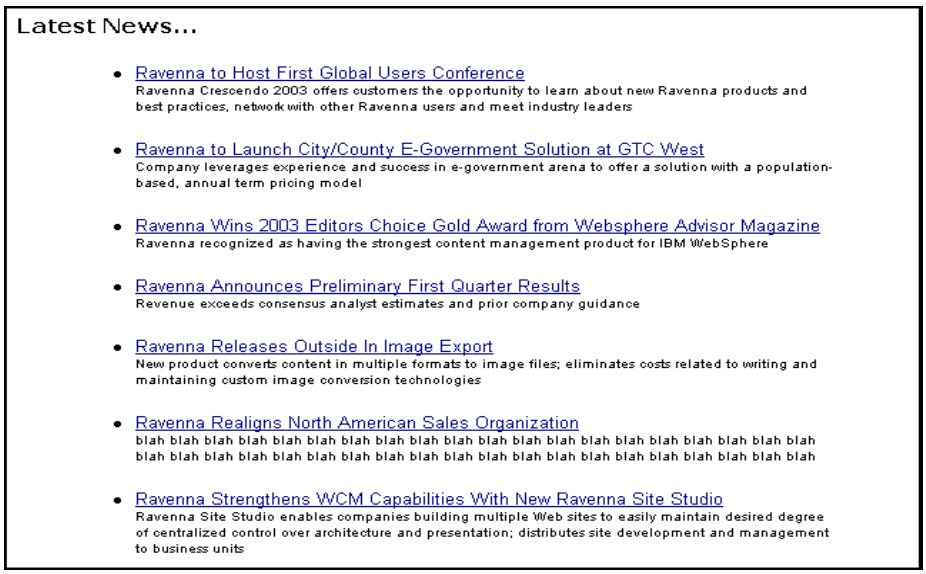
6.6.2 動的リスト

動的リストは、コンテンツ・サーバーにチェックインされた 1 つ以上のコンテンツ・アイテム（コントリビュータ・データ・ファイルまたはネイティブ・ドキュメント）の反復コンテンツを表します。リスト自体は、標準のコンテンツ・サーバー問合せに基づきます。動的リストのプレゼンテーションは、サーバー側スクリプトを含む動的リスト・フラグメント・スニペットで提供されます。このスクリプトは、検索結果の各行のループを実装しており、通常の Idoc メタデータ変数の構文を使用してメタデータ値を表示します。

問合せ自体は、Site Studio サービスの **SS_GET_SEARCH_RESULTS** を通じて実行されます。このサービスは、Content Server の GET_SEARCH_RESULTS コア・サービスとほぼ同じですが、一般的に使用されるロジックをカプセル化するいくつかの追加パラメータを使用します。また、SS_GET_SEARCH_RESULTS サービスは、（オプションで）レスポンスの ResultSet に **ssUrl** という追加列を提供します。この列には、データ・ファイルまたはネイティブ・ドキュメントの URL 計算を再利用するための 3 つの標準ルールに従って生成された、ResultSet の管理対象アイテムに対するパス・ベースの URL が含まれます。

動的リストの例

図 6-2 動的リストの例



この例では、news_story というタイプのドキュメントに対する問合せに基づいて、ニュース記事のリストが表示されています。

このプレゼンテーションを作成する場合、次のフラグメント・スニペットを使用できます。

```
<!--$ssQueryText= getValue("#active", ssFragmentInstanceId & "_ssQueryText")-->
<!--$ssSortField= getValue("#active", ssFragmentInstanceId & "_ssSortField")-->
<!--$ssSortOrder= getValue("#active", ssFragmentInstanceId & "_ssSortOrder")-->
<!--$ssResultCount= getValue("#active", ssFragmentInstanceId & "_ssResultCount")-->
<!--$ssLimitScope= getValue("#active", ssFragmentInstanceId & "_ssLimitScope")-->
<!--$ssTargetNodeId= getValue("#active", ssFragmentInstanceId & "_ssTargetNodeId")-->
<!--$ssNextRow= getValue("#active", ssFragmentInstanceId & "_NextRow")-->
<!--$ssUserSearchText= ""     -->
<!--$ssSourceSiteId= siteId -->
<!--$ssSourceNodeId= nodeId -->
<!--$ssWebsiteObjectType= ""     -->
<!--$ssDontShowInLists= "true" -->

<!--$if strEquals(ssNextRow, '')-->
  <!--$ssNextRow=1-->
<!--$endif-->
```

```
<!--$QueryText=eval(ssQueryText)-->
<!--$SortField=ssSortField-->
<!--$SortOrder=ssSortOrder-->
<!--$ResultCount=ssResultCount-->
<!--$StartRow=ssNextRow-->

<!--$executeService("SS_GET_SEARCH_RESULTS")-->

<!--$if SearchResults-->

   <!--$ssFirstHit=ssNextRow-->
   <!--$ssLastHit=ssNextRow + SearchResults.#numRows - 1-->

   <ul>
      <!--$loop SearchResults-->
         <li>
            <a class="rvh_newslist" href="<!--$ssUrl-->"><!--$dDocTitle--></a><br>
            <div class="rvh_newslist-comment"><!--$xComments--></div><br>
         </li>
      <!--$endloop-->
   </ul>

   <!--$ssNextRow = ssNextRow + ResultCount-->
   <!--$if ssNextRow le TotalRows-->
      <a class="rvh_newslist" href='?<!--$ssFragmentInstanceId-->_
NextRow=<!--$ssNextRow-->'>More...</a>
   <!--$endif-->

<!--$else-->
   <div class="rvh_newslist">
      No News Stories Available.
   </div>
<!--$endif-->
```

動的リストには、次の名前付きパラメータの定義が含まれる必要があります。

- **ssQueryText:** managedquery タイプのパラメータ。動的リストの検索基準を含みます。
- **ssSortField:** text タイプのパラメータ。動的リストの検索結果をソートするメタデータ・フィールドの名前を含みます。
- **ssSortOrder:** text タイプのパラメータ。動的リストの検索結果のソート順序を含みます。
- **ssResultCount:** リストに表示される結果の数。
- **ssLimitScope:** boolean タイプのパラメータ。検索基準を現在の Web サイトに制限するかどうかを示す true または false の値を含みます。

有効範囲制限ロジック

動的リストには、ssLimitScope というパラメータがあり、検索をこの Web サイトの一部である管理対象ドキュメントのみに制限するかどうかを指定する true または false の値を設定できます。

Site Studio リリース 6.5 では、この機能は xWebsiteID フィールドを使用して実装されていました。リリース 7.2 で、xWebsiteID フィールドが xWebsites フィールドに置き換えられたため（2-4 ページの「xWebsites」を参照）、動的リストの有効範囲制限ロジックも更新されました。

リリース 7.5.1 以上でも xWebsites フィールドは引き続き使用されますが、有効範囲制限ロジックは新規サービスの SS_GET_SEARCH_RESULTS 内にカプセル化されました。そのため、動的リスト・フラグメントは、標準の GET_SEARCH_RESULTS サービスのかわりにこの新規サービスをコールして問合せを実行するかぎり、個別にこのロジックを提供する必要はありません。

現在、有効範囲制限機能のロジックは次のように単純化されています。

```
<!--$ssLimitScope="true"-->
```

動的リストでの登録と除外

動的リストには、コントリビュータでリストを対象に登録と除外を行うことのできる機能もあります（『Oracle Universal Content Management Site Studio コントリビュータ・ガイド』を参照）。この場合、Web サイトのリストから除外されていないドキュメントのみに検索結果を制限する必要があります。前述のとおり、リリース 7.5.1 以上では、このロジックは SS_GET_SEARCH_RESULTS サービス内にカプセル化されているため、動的リスト・フラグメントで実装する必要があるのは、サーバー側変数の宣言のみです。

現在、リストに表示しないようにする機能のロジックは、次のように単純化されています。

```
<!--$ssDontShowInLists="true"-->
```

6.7 カスタム要素

カスタム要素は、ユーザー定義の要素です。他の要素（WYSIWYG、テキスト専用、イメージ専用、静的リスト、動的リスト）に加え、カスタム要素により、個々のビジネス・ニーズに合せて Site Studio を拡張する方法が提供されます。カスタム要素は、他の要素が適切ではないときに作成します。

カスタム要素は、基本的にコントリビュータ・フォームの IFRAME 内に存在する完全な HTML タイプのファイル（htm、hcsp、jsp など）です。カスタム要素が Site Studio 要素として正しく機能するためには、カスタム要素で API を使用して複数のコールバックを実装する必要があります。

重要： カスタム要素は、Site Studio 10*g*R3（10.1.3.3.3）で変更されています。以前のリリースの Site Studio サイトをアップグレードする場合、サイト内のすべてのカスタム要素を手動で更新する必要があります。6-15 ページの「下位互換性とアップグレード」を参照してください。

この項の内容は次のとおりです。

6.7.1 カスタム要素の作成

カスタム要素は、「Site Asset」ペインで作成するか、既存のフォームからコピーできます。

カスタム要素の作成

カスタム要素は、デザイナの「Site Assets」ウィンドウで次のように作成できます。

1. 「Site Assets」ウィンドウのプルダウン・メニュー内の**「Custom Element Form」**を選択します。
2. 「Create New」アイコンをクリックし、**「New」**→**「Custom Element」**を選択します。

   「Assign Metadata」画面が表示されます。
3. 「Assign Metadata」画面の適切なフィールドにデータを入力し、新規フォームをチェックインします。

既存のカスタム要素のコピー

フォームは、「Site Assets」ペインで次のようにコピーすることもできます。

1. 「Site Assets」ウィンドウのプルダウン・メニュー内の**「Custom Element Form」**を選択します。
2. 「Create New」アイコンをクリックし、**「Copy」→「from Server」**または**「from Local」**を選択します。
3. カスタム要素フォームを選択します。

   「Assign Metadata」ダイアログが表示されます。
4. 必要なメタデータを入力し、コンテンツ ID を変更します。新規フォームをチェックインします。
5. これで、フォームが「Site Assets」ペインのフォーム・リストに表示されます。フォームをダブルクリックすると編集できます。

6.7.2 カスタム要素の実装

カスタム要素がコントリビュータ・フォームで正しく機能するためには、カスタム要素で API を使用していくつかのコールバックを実装する必要があります。次の項では、カスタム要素を作成する観点から ElementAPI とそのメソッドを中心に説明します。

> **注意：** カスタム要素のパス（URL）は、コントリビュータ・フォームと同じドメインである必要があります。これにより、コントリビュータ・フォームとカスタム要素が、クロスドメイン・スクリプト・ルールに違反することなく通信できるようにします。

ElementAPI

ElementAPI オブジェクトは、コントリビュータ・フォームとカスタム要素間の通信を支援するカスタム要素ページに明示的にロードされる JavaScript オブジェクトです。ElementAPI により、カスタム要素がコントリビュータ・フォームと通信するためのメソッドと、コントリビュータ・フォームがカスタム要素に通知を送信するためのコールバック・メカニズムが提供されます。

ElementAPI JavaScript オブジェクトのロード

ElementAPI とそのサポート・ライブラリを使用する前に、ElementAPI をカスタム要素ページにロードする必要があります。ElementAPI のロード後、カスタム要素は、ページの初期化を継続し、カスタム要素がロードされて使用可能になったことをコントリビュータ・フォームに通知します。

```
<html>
<head>
  <title>Default Custom Element Form</title>
    <script type="text/javascript">
    function Initialize()
    {
     //============================================================================
     // TODO: ElementAPI is loaded. Place Custom Element initialization code here.
     //============================================================================
     // Let the Contributor Form know that this Custom Element is loaded and
     // ready.

    ElementAPI.Ready();
    }

     // Load the ElementAPI and its supporting libraries - then call Initialize()
     // Parameter 1: The Custom Element's window object. This parameter uniquely
     // identifies the Custom Element to the Contributor Form.
     // Parameter 2: A function pointer. This function will be executed after the
```

```
     // ElementAPI and its supporting libraries are loaded.

    try {
         window.top.WCM.InitializeCustomElement(window, Initialize);
    } catch(e) { }
    </script>
</head>
<body>
  <h3>Default Custom Element Form</h3>
</body>
</html>
```

注意： コード例では、ElementAPI をロードし、コントリビュータ・フォームに対して ElementAPI がロードされて使用可能になったことを通知します。ただし、データの収集または保存は行いません。

コントリビュータ・フォームからカスタム要素への通信

コントリビュータ・フォームは、カスタム要素により実装された関数を実行することでカスタム要素と通信します。初期化プロセスの一部として、カスタム要素では、コントリビュータ・フォームにこれらの関数のポインタを渡して各関数を登録する必要があります。

次の表に、コントリビュータ・フォームに登録できる関数を示します。これらの関数は、いずれもカスタム要素により実装する必要はありませんが、コントリビュータ・ユーザーからデータを収集して保存する場合は、いくつかの関数が必要になります。また、（IsDirty() 関数以外の）すべての関数は、実行されると、タスク完了時の実行のためにコールバック関数のポインタを渡します。これにより、実行時にカスタム要素で非同期タスクを実行する必要がある場合に、非同期通信を行うことができます。

関数シグネチャ	説明
CanCloseElement(callback);	コントリビュータ・フォームは、コントリビュータ・ユーザーが更新を実行したときにこのメソッドを実行します。関数の実装では、カスタム要素を安全に閉じることができるかどうかを評価する必要があります。たとえば、データが検証に合格しない場合、カスタム要素が閉じられないことを示す必要があります。
GetElementContent(callback);	コントリビュータ・フォームは、コントリビュータ・ユーザーが更新を実行したときにこのメソッドを実行します。関数の実装では、保存される文字列コンテンツを戻す必要があります。
Hide(callback);	コントリビュータ・フォームは、カスタム要素の上に HTML 要素をオーバーレイ表示する DHTML タスクをフォームで実行するときに常にこのメソッドを実行します。たとえば、このメソッドは、「Metadata」タブがアクティブ化され、コントリビュータ要素が隠蔽されたときに実行されます。 このメソッドは、特に Ephox ベースの要素向けに導入されています（Java アプレットには常に top z-index が含まれるため）。他のすべての要素（HTML ベースの要素）では、このメソッドを無視できます。
Show(callback);	コントリビュータ・フォームは、カスタム要素を再表示する（オーバーレイを削除する）DHTML タスクをフォームで実行するときに常にこのメソッドを実行します。 このメソッドは、特に Ephox ベースの要素向けに導入されています（Java アプレットには常に top z-index が含まれるため）。他のすべての要素（HTML ベースの要素）では、このメソッドを無視できます。

関数シグネチャ	説明
IsDirty();	コントリビュータ・フォームは、更新なしにフォーム・ポップアップが閉じられるときに常にこのメソッドを実行します。カスタム要素は、未保存の変更があるかどうかを評価して、保存していない変更がある場合はコントリビュータ・ユーザーに通知します。

次に、カスタム要素がコントリビュータ・フォームに関数を登録する方法を示すJavaScriptのコード例を示します。

```
function CanCloseElement(callback)
{
    // No data validation in this sample - just pass back a true value.
    callback({canClose: true});

    // Here is an example of passing a false value
    // callback({canClose: false, reason: 'Failed validation. Use only lowercase
    // letters.'};
}

function GetElementContent(callback)
{
    // Pass back some sample content for demo purposes.
    callback('This is my Custom Element Content.');
}

function Show(callback)
{
    // Just handle this notification by executing the callback.
    callback();
}

function Hide(callback)
{
    // Just handle this notification by executing the callback.
    callback();
}

function IsDirty()
{
    // This Custom Element is never dirty - so pass a false value.
    return {isDirty: false};
}

// Set callback methods for the Contributor Form to send notifications to this
// Element.
ElementAPI.SetCallback('CanCloseElement', CanCloseElement);
ElementAPI.SetCallback('GetElementContent', GetElementContent);
ElementAPI.SetCallback('Show', Show);
ElementAPI.SetCallback('Hide', Hide);
ElementAPI.SetCallback('IsDirty', IsDirty);
```

カスタム要素からコントリビュータ・フォームへの通信

カスタム要素では、ElementAPI JavaScript オブジェクトを使用してコントリビュータ・フォームとの通信を開始します。次に、使用可能な ElementAPI メソッドのリストを示します。（使用例は、6-15 ページの「サンプル・カスタム要素」に説明されている ss_sample_form.htm を参照してください。）

関数シグネチャ	説明
ElementAPI.GetDefaultData();	データ・ファイルに格納されたデフォルト・コンテンツを取得します。
ElementAPI.GetSearchResults(options);	Content Server の「Get Search Results」ページを表示します。
ElementAPI.GetQueryText(options);	「Get Query Text」ユーザー・インタフェースを表示します。
ElementAPI.CaptureQuery(options);	Content Server の「Capture Query」ページを表示します。
ElementAPI.GetHyperlink(options);	「Hyperlink Wizard」ユーザー・インタフェースを表示します。
ElementAPI.FocusForm(options);	親ウィンドウにフォーカスし、要素ウィンドウを隠蔽します。
ElementAPI.SetHostHeight(height);	要素を格納する IFRAME の高さを設定します。
ElementAPI.SetRequiredIndicator(isRequired);	「Contributor Form」ユーザー・インタフェースで「Required」グラフィック・インジケータを切り替えます。
ElementAPI.GetSite(options);	「Choose Website」選択ユーザー・インタフェースを表示します。
ElementAPI.GetSection(options);	「Choose Website Section」選択ユーザー・インタフェースを表示します。
ElementAPI.GetColor(options);	「Color」選択ユーザー・インタフェースを表示します。
ElementAPI.GetFont(options);	「Get Font」選択ユーザー・インタフェースを表示します。

ElementAPI 依存スクリプト

ElementAPI がカスタム要素ページにロードされる際に、ElementAPI 依存スクリプトもロードされます。これらのスクリプトには、JavaScript WCM ライブラリのほとんどが含まれており、カスタム要素の作成者が使用することもできます。次のスクリプト・ファイルがカスタム要素にロードされます。

- wcm.js
- ./base/wcm.dhtml.js
- ./base/wcm.get.js
- ./base/wcm.http.js
- ./base/wcm.popup.js
- ./sitestudio/wcm.contentserver.popup.js
- ./form/elements/wcm.elementapi.js
- ./sitestudio/elements/wcm.sitestudio.elementapi.js
- ./sitestudio/wcm.idc.js
- ./form/elements/element/wcm.element.js
- ./form/elements/custom/wcm.custom.js

カスタム要素のソース・モード・タイプの例

次の単純なカスタム要素の例では、コントリビュータ・ユーザーにより TEXTAREA に入力されたコンテンツを保存し、それをレイアウトに直接表示します。

```
<html>
<head>
   <title>Default Custom Element Form</title>
   <script type="text/javascript">
   function CanCloseElement(callback)
   {
      // No data validation in this sample - just pass back a true value.
      callback({canClose: true});

      // Here is an example of passing a false value
      // callback({canClose: false, reason: 'Failed validation. Use only lowercase
      // letters.'};
   }
   function GetElementContent(callback)
   {
      // Pass back the user edits
      callback(document.getElementById('source').value);
   }

   function Show(callback)
   {
      // Just handle this notification by executing the callback.
      callback();
   }

   function Hide(callback)
   {
      // Just handle this notification by executing the callback.
      callback();
   }

   function IsDirty()
   {
      // Test for a dirty state by comparing the the default value with the value
      // in the TEXTAREA.
      return {isDirty: (document.getElementById('source').value !==
ElementAPI.GetDefaultData())};
   }

   function Initialize()
   {
      // Set callback methods for the Contributor Form to send notifications to
      // this element.
      ElementAPI.SetCallback('CanCloseElement', CanCloseElement);
      ElementAPI.SetCallback('GetElementContent', GetElementContent);
      ElementAPI.SetCallback('Show', Show);
      ElementAPI.SetCallback('Hide', Hide);
      ElementAPI.SetCallback('IsDirty', IsDirty);

      // Set the default data
     document.getElementById('source').value = ElementAPI.GetDefaultData() || '';

      // Let the Contributor Form know that this Custom Element is loaded and
      // ready.
      ElementAPI.Ready();
   }

   // Load the ElementAPI and its supporting libraries - then call Initialize()
   // Parameter 1: The Custom Element's window object. This parameter uniquely
```

```
   // identifies the Custom Element to the Contributor Form.
   // Parameter 2: A function pointer. This function will be executed after the
   // ElementAPI and its supporting libraries are loaded.
   try {
      window.top.WCM.InitializeCustomElement(window, Initialize);
   } catch(e) { }
   </script>
</head>
<body>
   <textarea id="source" rows="10" style="width:100%"></textarea>
</body>
</html>
```

6.7.3 カスタム要素のテスト

カスタム要素をテストするのに最善の方法は、カスタム要素をレイアウトまたはリージョンに追加して、そのリージョンをコントリビュータ・フォームで編集することです。コントリビュータ・フォームが開いたら、作成者は独自のカスタム要素をテストできます。

1. デザイナ・アプリケーション内で、リージョンを含むレイアウトを作成します。
2. カスタム要素をリージョンに追加します。
3. 新規作成されたカスタム要素フォームを参照するようにカスタム要素を構成します。
   a. カスタム要素をダブルクリックして、要素構成ダイアログを表示します（まだ表示されていない場合）。
   b. 「Settings」ボタンをクリックして、「Custom Element Settings」ダイアログを表示します。
   c. 「Form Url」フィールドにデータを入力するため、省略記号をクリックして問合せ選択ダイアログを表示し、新規作成されたカスタム要素フォームを選択します（前の項を参照）。
4. リージョン・コンテンツをリージョンに割り当てます（まだ割り当てられていない場合）。
5. レイアウトを保存します。
6. カスタム要素をテストします。
   a. デザイナ・ツールバーで「Preview in Browser」を選択します。
   b. ブラウザで、割当て済のキーの組合せを使用して（デフォルトは [Ctrl]+[Shift]+[F5]）、コントリビューション・モードに移行します。
   c. ブラウザで、リージョンのポップアップ・メニューから「Edit」を選択し、コントリビュータ・フォームを起動します。

この時点で、デザイナ・アプリケーションとコントリビュータ・フォームが両方とも開いている場合、作成者は、デザイナ・アプリケーションでカスタム要素の変更を保存し、コントリビュータ・フォームをリフレッシュしてカスタム要素に加えた新しい変更をテストできます。

6.7.4 カスタム要素のデバッグ

JavaScript デバッガや従来の JavaScript のデバッグ方法を使用する以外に、Site Studio コンソール・ウィンドウも便利なデバッグ・ツールとして使用できます。コンソール・ウィンドウは、実行時情報の提供、コンテキスト固有のエラーの格納、および「Immediate」ウィンドウなどのデバッグ機能の提供を行う Site Studio コントリビュータのロギング・メカニズムです。たとえば、カスタム要素がメソッドを実装していない場合や、コールバック・メソッドを適切に登録していない場合、コンソール・ウィンドウに警告が表示されます。

6.7.5 下位互換性とアップグレード

すべてのレガシー・カスタム要素フォーム（つまり、以前のリリースの Site Studio と互換性のあるカスタム要素フォーム）は、Site Studio 10*g*R3（10.1.3.3.3）とは互換性がないため、手動でアップグレードする（再作成する）必要があります。下位互換性が維持されていない主な理由は、以前の Site Studio が Internet Explorer の独自仕様の window.external 機能に依存しているためです。window.external 機能は、コード実行の時点でブロックされ、クロスブラウザ / プラットフォームの DHTML ソリューションでは簡単に複製できません。下位互換性をなくしたことによるメリットは、新規カスタム要素がより一層柔軟になり、（クロスブラウザ / プラットフォーム・ソリューションであることに加え）コントリビュータ・アプリケーション・アーキテクチャに緊密に統合されたことです。

レガシー・カスタム要素フォームの検出

レガシー・カスタム要素フォームは、新しいコントリビュータ・アプリケーションにロードされると、デフォルトで検出されます。その際、コントリビュータ・フォーム内の検出された場所にエラー・メッセージが表示されます。コントリビュータ・アプリケーションは、この検出を行うために、まずカスタム要素フォームをダウンロードし、ソース・コードを解析して、カスタム要素フォームが新しいコントリビュータ・アプリケーションと互換性があるかどうかを確認します。

レガシー・カスタム要素を検出する機能とそのオーバーヘッドは、本番インストール環境では不要です。すべてのレガシー・カスタム要素フォームをアップグレードしたら、この機能は無効にする必要があります。レガシー・カスタム要素フォームの検出機能を無効化するには、次の行を Content Server の config.cfg ファイルに追加して、サーバーを再起動してください。

```
SSValidateCustomElements=false
```

6.7.6 サンプル・カスタム要素

Site Studio 10*g*R3（10.1.3.3.3）には、ss_simple_form.htm、ss_sample_form.htm および ss_flash_form.htm という 3 つのサンプル・カスタム要素が付属しています。これらのサンプル・フォームは、コントリビュータ・フォームのフレームワーク内で動作するカスタム要素の実装例です。

これらのサンプルは、Site Studio コンポーネントのインストール時に、Custom Element Form という xWebsiteObjectType メタデータ値とともにコンテンツ・サーバーにチェックインされます。以前のリリースのサンプル・カスタム要素フォームの旧バージョンについては、新規バージョンとしてチェックインされる更新バージョンが存在します。

ss_simple_form.htm

このサンプル・フォームは、最小限のコードを含む単純なカスタム要素の例です。この特定の例では、TEXTAREA HTML 要素を使用してテキスト専用のカスタム要素を実装しています。

ss_sample_form.htm

このサンプル・フォームは、単純なテキスト専用フィールドの実装に加え、すべての ElementAPI コールを具体的に示したカスタム要素の例です。

ss_flash_form.htm

このサンプル・フォームは、コントリビュータ・ユーザーがコンテンツ・サーバーから FLASH オブジェクトを選択できるカスタム要素の例です。

7

Idoc スクリプト拡張機能

Site Studio では、Web サイトを実行するために複数の Idoc スクリプト拡張機能を使用します。

注意： これらのスクリプト拡張機能は、Site Studio の各リリースで変更される可能性があります。

7.1 ssIncludeXml()

このスクリプト拡張機能は、管理対象 XML ファイル内の任意の要素を抽出して Idoc 文字列変数に戻すことができる Site Studio のコア・メソッドです。Idoc 文字列変数は、HTML スニペットとして Web ページ上に直接配置できます。抽出される XML ノードのコンテンツは、現在のレイアウト・ページの有効範囲で詳細に評価され、必要に応じて追加のサーバー側 Idoc スクリプトがインクルードされます。

パラメータは次のとおりです。

- XML データ・ファイルの dDocName
- XPath 式（XML データ・ファイルの 1 つ以上の特定ノードを識別する式）

7.2 ssGetXmlNodeCount()

このスクリプト拡張機能は、管理対象 XML ファイルにおける特定の要素の反復インスタンスの数を戻します。この拡張機能は、反復コントリビューション要素のセットを表示するために、静的リスト・フラグメントの実装で主に使用されます。

パラメータは次のとおりです。

- XML データ・ファイルの dDocName
- XPath 式（XML データ・ファイルの 1 つ以上の特定ノードを識別する式）

7.3 ssIncDynamicConversion()

このスクリプト拡張機能は、ネイティブ・ドキュメントの動的変換を実行するために使用できます。この変換で生成される HTML は、HTML スニペットの形式で Web ページ上に直接配置できる Idoc 文字列変数に戻されます。このメソッドでは、コール元は、使用する管理対象変換テンプレートおよび変換レイアウトを指定できます。

パラメータは次のとおりです。

- 変換するネイティブ・ドキュメントの dDocName
- 変換に使用するテンプレートの dDocName
- 変換に使用するレイアウト・テンプレートの dDocName

7.4 ssIncDynamicConversionByRule()

このスクリプト拡張機能は、ssIncDynamicConversion() 拡張機能と同様に、ネイティブ・ドキュメントの動的変換を実行するために使用できます。この変換で生成される HTML は、HTML スニペットの形式で Web ページ上に直接配置できる Idoc 文字列変数に戻されます。

ssIncDynamicConversion とは異なり、この拡張機能では、使用する変換テンプレートを決定するために Dynamic Converter ルール・エンジンで指定されたルールを使用します。したがって、コール元は、管理対象変換テンプレートまたは変換レイアウトを指定する必要がありません。

パラメータは次のとおりです。

- 変換するネイティブ・ドキュメントの dDocName
- 使用する Dynamic Converter ルールの名前

7.5 ssIncDynamicConversionByRulesEngine()

このスクリプト拡張機能は、ssIncDynamicConversion() 拡張機能と同様に、ネイティブ・ドキュメントの動的変換を実行するために使用できます。この変換で生成される HTML は、HTML スニペットの形式で Web ページ上に直接配置できる Idoc 文字列変数に戻されます。

ssIncDynamicConversion とは異なり、この拡張機能では、使用する変換テンプレートを決定するために標準の Dynamic Converter ルール・エンジンを使用します。したがって、コール元は、管理対象変換テンプレートまたは変換レイアウトを指定する必要がありません。

パラメータは次のとおりです。

- 変換するネイティブ・ドキュメントの dDocName

7.6 ssIncInlineDynamicConversion()

このスクリプト拡張機能は、ssIncDynamicConversion() 拡張機能と同様に、ネイティブ・ドキュメントの動的変換を実行するために使用できます。この変換で生成される HTML は、HTML スニペットの形式で Web ページ上に直接配置できる Idoc 文字列変数に戻されます。ssIncDynamicConversion とは異なり、この拡張機能では、動的変換用に空白の組込みテンプレートを使用するため、コール元は管理対象変換テンプレートまたは変換レイアウトを指定する必要がありません。

パラメータは次のとおりです。

- 変換するネイティブ・ドキュメントの dDocName

7.7 ssIsNativeDoc()

このスクリプト拡張機能は、管理対象ドキュメントが、動的変換を実行できるネイティブ・ドキュメントであるか、変換の不要なコントリビュータ・データ・ファイルであるかを検出するために使用できます。

パラメータは次のとおりです。

- テストするドキュメントの dDocName

7.8 ssCheckAccessPrepareMenu()

この関数は、2 つの処理を実行します。まず、ユーザーがファイルへの適切なアクセス権を持っているかどうかをチェックします。次に、ページがコントリビュータ・モードで表示された場合にコントリビューション・アイコンに関連付けるポップアップ・メニューを作成します。ポップアップ・メニューの各アクションは、アクション文字列でリクエストされると、メニューに追加される前に十分なユーザー・アクセス権があるかどうかをチェックされます。

パラメータは次のとおりです。

- データ・ファイルの dDocName（ポップアップ・メニューの準備）
- アクション文字列（設計者が作成）

7.9 ssRandom()

このスクリプト拡張機能は、ランダム数値を生成するために使用します。パラメータはありません。

7.10 ssGetNodeProperty()

このスクリプト拡張機能は、現在のセクションまたは明示的に指定したセクションのノード・プロパティを取得するために使用できます。通常は、カスタム・ノード・プロパティへのアクセスに使用します。

パラメータは次のとおりです。

- 問い合せるセクションの nodeId（オプション）
- プロパティ名

たとえば、次のコマンドでは、現在のセクションのセクション・ラベルを取得します。

```
<!--$ssGetNodeProperty("label") -->
```

また、次のコマンドでは、セクション 474 のセクション・ラベルを取得します。

```
<!--$ssGetNodeProperty("474", "label") -->
```

7.11 ssGetWebsiteNodeType()

このスクリプト拡張機能は、サイト・ノード（階層のルート）と通常のセクション・ノードの違いを判別するために使用できます。また、ASP サイトと ASP 以外のサイト（またはセクション・ノード）の違いも判別できます。

パラメータは次のとおりです。

- nodeId

戻り値は次のいずれかです。

- 1: HCSP/JSP タイプのサイトのサイト・ノード
- 2: HCSP/JSP タイプのサイトのセクション・ノード
- 101: ASP タイプのサイトのサイト・ノード
- 102: ASP タイプのサイトのセクション・ノード

7.12 ssGetCoreMajorVersion()

このスクリプト拡張機能は、基礎となるコンテンツ・サーバーのメジャー・バージョンを取得するために使用できます。パラメータはありません。

7.13 ssSplitString()

このスクリプト拡張機能は、指定したデリミタに基づいて文字列をセグメントに分割するために使用できます。

パラメータは次のとおりです。

- 分割する文字列
- 文字列の分割に使用するデリミタ（「,」など）
- 分割操作の結果を格納する結果セットの名前

戻り値は、指定した文字列から切り出されたセグメントの数を示す整数です。この拡張機能では、分割文字列をリストした String という名前の 1 つの列を持つ ResultSet も生成されます。

7.14 ssGetWebsiteName()

このスクリプト拡張機能は、コンテンツ・サーバーで任意の Web サイトの Web サイト名を取得するために使用できます。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子

7.15 ssGetSiteProperty()

このスクリプト拡張機能は、コンテンツ・サーバーで任意の Web サイトのサイト・プロパティを取得するために使用できます。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子
- **propertyName:** 必要な値を保持するプロパティの名前

7.16 ssGetFirstNodeId()

このスクリプト拡張機能は、コンテンツ・サーバーで任意の Web サイトのルート・ノードの識別子（nodeId）を取得するために使用できます。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子

7.17 ssGetRelativeNodeId()

このスクリプト拡張機能は、特定のノードを基準としてノードの識別子（nodeId）を取得するために使用できます。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子。
- **nodeId:** コンテキスト・ノードの一意の識別子。
- **relative:** 関係として次のいずれかを指定する必要があります。
  - child
  - parent
  - prior
  - next

7.18 ssLoadSiteNavResultSet()

このスクリプト拡張機能は、コンテンツ・サーバーの任意の Web サイトに対応するアクティブ階層を含む SiteStudioNavNodes という ResultSet を、Idoc 拡張機能環境に作成するために使用できます。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子。

SiteStudioNavNodes ResultSet には、次の 5 つの列があります。

- **nodeId:** ノードの一意の識別子。
- **parentNodeId:** 親ノードの一意の識別子。
- **label:** ノードのラベル。
- **level:** サイト階層におけるノードの深さ。ルート・セクションのレベルは 0（ゼロ）です。
- **href:** ノードのプライマリ・ページに対するパス・ベースのサイト相対 URL。

7.19 ssGetServerRelativeUrl()

このスクリプト拡張機能は、指定したノードのプライマリ・レイアウトに対するサーバー相対 URL を生成するために使用できます。この相対 URL には、末尾の urlPageName（通常は index.htm）が含まれます。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子
- **nodeId:** ノードの一意の識別子

7.20 ssGetServerRelativePath()

このスクリプト拡張機能は、指定したノードのプライマリ・レイアウトに対するサーバー相対 URL を生成するために使用できます。この相対 URL には、末尾の urlPageName（通常は index.htm）は含まれません。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子
- **nodeId:** ノードの一意の識別子

7.21 ssGetUrlPageName()

このスクリプト拡張機能は、指定したノードの urlPageName（通常は index.htm）を取得するために使用できます。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子
- **nodeId:** ノードの一意の識別子

7.22 ssGetNodeLabel()

このスクリプト拡張機能は、指定したノードのラベルを取得するために使用できます。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子
- **nodeId:** ノードの一意の識別子

7.23 ssGetNodeLabelPath()

このスクリプト拡張機能は、GUI ラベルに使用できる、Web サイトのルートを基準とする「/」区切りの文字列（/products/servers/web server など）を取得するために使用できます。

パラメータは次のとおりです。

- **siteId:** Web サイトの一意の識別子
- **nodeId:** ノードの一意の識別子

7.24 ssGetAllSites()

このスクリプト拡張機能は、使用可能なすべての Web サイトのリストを含む ResultSet を生成するために使用できます。

パラメータは次のとおりです。

- **resultSetName:** 生成する ResultSet の名前

7.25 ssLink()

このスクリプト拡張機能は、名前付きドキュメントに対するパス・ベースの URL を生成するために使用します。

パラメータは次のとおりです。

- **dDocName:** 管理対象アイテムの dDocName
- **targetNodeId:** ターゲット・コンテキストとして使用するノードの一意の識別子（オプション）
- **targetSiteId:** ターゲット・コンテキストとして使用する Web サイトの一意の識別子（オプション）

7.26 ssNodeLink()

このスクリプト拡張機能は、指定したノードに対するパス・ベースの URL を生成するために使用します。

パラメータは次のとおりです。

- **nodeId:** ノードの一意の識別子
- **siteId:** ノードを含む Web サイトの一意の識別子（オプション）

7.27 ssLocalizeMessage()

このスクリプト拡張機能は、名前付き文字列リソースのローカライズされたバージョンを取得するために使用します。

パラメータは次のとおりです。

- **localToken:** 現在のユーザーの言語における検索トークンに対応する一意の名前

7.28 ssWeblayoutUrl()

このスクリプト拡張機能は、パスまたは dDocName からファイルの完全な Web アドレスを特定するために使用します。この拡張機能は、通常、データ・ファイルのイメージへのパスに対して使用されます。

パラメータは次のとおりです。

- **targetId:** パスまたは dDocName の値

8

Idoc スクリプト変数

Site Studio では、Web サイトを実行するために複数の Idoc 変数を使用します。

注意： これらの変数は、Site Studio の各リリースで変更される可能性があります。

8.1 HttpWebsitesRoot

Site Studio Websites フォルダの HTTP による絶対位置を定義します。一般的な場所は次のとおりです（server はマシン名で、instance はコンテンツ・サーバーのインスタンス名です）。

```
http://server/instance/websites/
```

8.2 HttpRelativeWebsitesRoot

Site Studio Websites フォルダの HTTP による相対位置を定義します。一般的な場所は次のとおりです（instance はコンテンツ・サーバーのインスタンス名です）。

```
instance/websites/
```

8.3 HttpFragmentsRoot

Site Studio fragments フォルダの HTTP による絶対位置を定義します。一般的な場所は次のとおりです（server はマシン名で、instance はコンテンツ・サーバーのインスタンス名です）。

```
http://server/instance/fragments/
```

8.4 HttpRelativeFragmentsRoot

Site Studio fragments フォルダの HTTP による相対位置を定義します。一般的な場所は次のとおりです（instance はコンテンツ・サーバーのインスタンス名です）。

```
instance/fragments/
```

8.5 SS_SERVER_NAME

現在のマシンのプロトコル、サーバー名およびポート番号（該当する場合）を定義します。たとえば、次のようになります（server はサーバー名です）。

```
http://server
http://server:81
https://server
```

8.6 HttpASPPath

標準の Site Studio get_page.asp ファイルに対する相対パスを定義します。

8.7 ssServerRelativeSiteRoot

サーバーを基準とした現在の Web サイトに対応するパス・ベースの URL のルート部分を定義します。次の値のいずれかが含まれます。

- **/:** 現在の Web サイトがフレンドリ・ドメイン・アドレス指定モードを使用してアクセスされる場合
- **/siteId/:** 現在の Web サイトがフレンドリ・フォルダ・アドレス指定モードを使用してアクセスされる場合

値が存在する場合、ルート・ノードの urlDirName により siteId は上書きされます。

9

Site Studio サービス

Site Studio には、Content Server の多くの新規サービスが導入されています。これらのサービスは、特に Web サイトの実行に使用されます。最も重要なサービスは次のとおりです。

注意： これらのサービスは、Site Studio デザイナ・アプリケーションおよびコントリビュータ・アプリケーションと、Site Studio コンポーネントにより使用されます。これらのサービスは、Site Studio アプリケーションの各リリースで変更される可能性があります。

9.1 SS_GET_PAGE

コア・サービスである SS_GET_PAGE は、特に Site Studio 用に作成されています。このサービスは、多くの構成要素から単一の Web ページを動的に生成することで、製品の中心的な役割を果します。

Site Studio リリース 7.2.1 以下では、SS_GET_PAGE サービスは、Site Studio Web ページの URL を直接参照するために使用されていました。Site Studio の現在のリリースでは、SS_GET_PAGE サービスは、Web ページの URL で直接使用されることはほとんどありません。かわりに、Web ページは、より標準的なパス・ベースの URL を使用してアクセスされます。（2-4 ページの「パス・ベースの URL」を参照してください。）

たとえば、次のようになります。

```
http://www.oracle.com/products/contentserver/index.htm
```

ただし、SS_GET_PAGE コア・サービス・コールは、これらの URL の背後で処理を実行するために依然として存在しています。また、現在は Web サーバーのフィルタ・プラグインがすべてのパス・ベースの Site Studio 関連の URL を中間で取得して、基礎となる SS_GET_PAGE サービス・コールに渡す前にそれらを複数の構成要素に変換しています。そのため、ここでは引き続きこのコア・サービスについて説明します。

SS_GET_PAGE サービスは、問合せ文字列の一部として、相互に排他的な次のパラメータのいずれかを必要とします。

- **nodeId:** 指定されると、サービスはサイト階層の該当ノードを検索し、そのノードの primaryUrl プロパティを取得します。これにより、ノードのプライマリ・ページを直接提供するのに十分な情報を取得できます。

  かわりに、URL で useSecondary=true というパラメータを指定して、ノードのセカンダリ・ページを取得することもできます。この場合、サービスはノードの secondaryUrl プロパティを取得し、セカンダリ・ページを提供します。この機能は、主にデザイナ・アプリケーションで使用されます。

- **ssDocName:**（有効な dDocName 値で）指定されると、サービスは管理対象コンテンツ・アイテムを検索し、3 つのルールによる評価（次の「コンテンツを再利用するための 3 つのルール」を参照）を使用して、コンテンツ・アイテムを表示する Web サイト・セクションを決定します。

  次に、このサービスは、セクションの primaryUrl および secondaryUrl プロパティを解析して、セクションのプライマリ・ページとセカンダリ・ページのどちらを表示するかを決定するための処理決定プロセスを実行します。決定プロセスが完了すると、適切なページが表示されます。

Site Studio デザイナのインタフェースとデザイナ・ガイドでは、サイト階層は Web サイトを構成するセクションを持つものとして説明されています。セクションは、製品の最初の開発時にはノードと呼ばれていました。そのため、SS_GET_PAGE サービスなどの多くの内部機能では、現在もそのように呼ばれています。Site Studio の用語では、ノードはセクションと同じです。

コンテンツを再利用するための 3 つのルール

nodeId パラメータとともに SS_GET_PAGE が使用される場合、サービスは指定したセクションのプライマリ・ページを表示するように明示的に指示されるため、追加の決定処理は不要です。ただし、**ssDocName** パラメータとともに SS_GET_PAGE が使用される場合、サービスはドキュメントを表示するセクションを決定する必要があります。これは、次の 3 つのルールにより決定されます。

1. **ターゲット :** オプションの **ssTargetNodeId** パラメータが URL に指定されている場合、そのセクションに管理対象ドキュメントが明示的に表示されます。
2. **セクション :** 管理対象ドキュメントの ***x*WebsiteSection** メタデータ・フィールドに値が含まれる場合、その値がセクションとして使用され、そのセクションに管理対象ドキュメントが表示されます。
3. **ソース :** オプションの **ssSourceNodeId** パラメータが現在のページのノードとして URL に指定されている場合、そのセクションに管理対象ドキュメントが明示的に表示されます。

これらすべての処理により、コントリビュータは、同じサイトの異なるセクションで、または同じコンテンツ・サーバーの異なるサイトでコンテンツ（コントリビュータ・データ・ファイルとネイティブ・ドキュメント）を共有および再利用できます（デザイナ・ガイドのターゲット・セクションを使用したリージョン・コンテンツの共有に関する項を参照）。

（ssDocName パラメータを使用する）SS_GET_PAGE サービスは、管理対象ドキュメントを表示するセクションを決定したら、次にプライマリ・レイアウト・ページとセカンダリ・レイアウト・ページのどちらを表示するかを決定する必要があります。これは、次のアルゴリズムにより決定されます。

条件	アクション
ssDocName がセクションの primaryUrl プロパティのリージョン・パラメータと一致する場合	プライマリ・ページが表示されます。
ssDocName がセクションの secondaryUrl プロパティのリージョン・パラメータと一致する場合	セカンダリ・ページが表示されます。
それ以外の場合	セカンダリ・ページが表示されますが、REPLACEABLE リージョン・パラメータは ssDocName 値で置き換えられます。 REPLACEABLE リージョンは、セクションの secondaryUrlVariableField プロパティにより定義されます。

前述の説明は、ssDocName パラメータがコントリビュータ・データ・ファイルまたはネイティブ・ドキュメントを示していることが前提です。ssDocName パラメータがレイアウト・ファイルを示している場合、異なる処理決定プロセスが発生します。この場合、Site Studio は、そのレイアウトを使用しているセクションを特定するまでサイト階層を順番に検索します。ただし、管理対象レイアウトに対する直接リンクはほとんど使用されません。この方法は、通常、デザイナにより内部的に使用されます。

nodeId、ssDocName、ssTargetNodeId および ssSourceNodeId に加え、SS_GET_PAGE サービスでは、次のオプションの URL パラメータまたは Cookie の値も認識されます。

- **SSContributor:** このパラメータには、コントリビューション・モードで Web ページを表示するかどうかを示す true または false の値が含まれます。コントリビューション・モードでは、Web ページのコントリビューション・リージョンごとに、コンテンツ・アイテムのワークフロー・バージョンとコントリビューション・アイコンが表示されます。必要に応じて、このパラメータによりログイン・プロンプトも表示されます。URL パラメータとして渡されると、この値は自動的に Cookie 値として設定され、コントリビュータがサイトをナビゲートするのに合せて自動的に伝播されます。
- **PreviewId:** このパラメータは、Web ページの最新のチェックイン・バージョンのかわりに、一時的なプレビュー・バージョンを表示することを示します。このパラメータは、レイアウト・ページやコントリビュータ・データ・ファイルの新規バージョンをチェックインすることなくプレビュー・サービスを提供するため、(SS_GET_PAGE サービスと組み合せて）デザイナおよびコントリビュータ・アプリケーションでのみ使用します。URL パラメータとして渡されると、この値も自動的に Cookie 値として設定され、コントリビュータや設計者がサイトをナビゲートするのに合せてプレビュー・モードを維持するために自動的に伝播されます。

  このパラメータを直接使用することは避けてください。

エラー・ハンドラ・セクション

エラーは、SS_GET_PAGE サービス・コールの実行中のどの時点でも発生する可能性があります。たとえば、必要な REPLACEABLE リージョンが特定のセクションに対して指定されていない場合、エラーが発生することがあります。通常、これらのタイプのエラーが発生すると、コンテンツ・サーバーの標準のエラー・ページが表示され、コンシューマは Web サイトのコンテキストの外部に移されます。

エラーを表示しながらコンシューマを Web サイトのコンテキスト内にとどめるため、デザイナ・アプリケーションでは、サイト内の 1 つのセクションをエラー・ハンドラ・セクションとして指定できます。SS_GET_PAGE サービス内でエラーが発生すると、コンシューマは、エラー・ハンドラ・セクションに関連付けられたプライマリ・レイアウト・ページにリダイレクトされます。エラー情報を表示するレイアウト・ページでは、次の 2 つの Idoc 変数を使用できます。

- **ssErrorMessage:** 発生したエラーのテキスト説明
- **ssErrorCode:** 発生したエラーのタイプを示す数値エラー・コード（これにより、独自のエラーの説明を表示できます）

内部的には、エラー・ハンドラ・セクションを設定すると、サイト・プロパティの errorNodeId が、エラー・ハンドラ・セクションの nodeId を含むように設定されます。

Site Studio で出現する可能性のあるエラー・コードのリストは、次のとおりです。

ssErrorCode	ssErrorMessage
-0x100	"No Layout page specified for this part of the web site."
-0x101	"Failed to locate document information for document with content ID"
-0x102	"Document with Content ID '{1}' does not match the Primary or Secondary Url at section '{2}' (Id={3}) and there is no Replaceable Region defined."
-0x103	"Link to Section '{1}' (Id={2}) failed because there is no Primary URL defined for the section."
-0x104	"Link to Section '{1}' (Id={2}) failed because there is no Secondary URL defined for the section."
-0x105	"The Section '{1}' (Id={2}) is not part of a Site Studio web site."
-0x106	"Layout with Content ID '{1}' is not filed in a Site Studio web site."
-0x107	"The Layout with Content ID '{1}' was not found in any section Url of any web site."
-0x108	"Unable to identify in which web site section to display document with Content ID '{1}'."
-0x200	"An unknown error has occurred in the SS_GET_PAGE service call."

- **説明 :** Site Studio Web ページを表示します。
- **パラメータ :**

  セクションのプライマリ（スプラッシュ）・ページを表示するためのパラメータ :

  - **siteId:** サイトの識別子（オプション）。指定されない場合、この値は計算で求められます。
  - **nodeId:** セクションの識別子（必須）。

  セクションに特定のドキュメントを表示するためのパラメータ :

  - **ssDocName:** 表示するドキュメントの dDocName（必須）。
  - **ssTargetNodeId:** ドキュメントを表示するセクションの識別子（オプション）。
  - **ssTargetSiteId:** 指定された ssTargetNodeId を含むサイトの識別子（オプション）。指定されない場合、この値は計算で求められます。
  - **ssSourceNodeId:** ドキュメントの xWebsiteSection 値を使用できない場合にドキュメントを表示するセクションの識別子（オプション）。
  - **ssSourceSiteId:** 指定された ssSourceNodeId を含むサイトの識別子（オプション）。指定されない場合、この値は計算で求められます。
  - **SSContributor:** ページをコントリビューション・モードで提供するかどうかを指定します（オプション）。使用可能な値は、true と false です。
- **戻り値 :** Site Studio Web ページ。
- **セキュリティ :** ユーザーは、Web サイト・ページを構成するドキュメントに対して少なくとも読取りアクセス権を持っている必要があります。

9.2 SS_GET_ALL_SITES_EX2

- **説明 :** コンテンツ・サーバー上のすべてのサイトのリストを戻します。
- **パラメータ :** なし。
- **戻り値 :**
  - **numSites**=N
  - **site0**=*siteId*

    . . .
  - **siteN-1**=*siteId*

  SiteIds 結果セットに同じ情報が戻されます。
- **セキュリティ :** 戻されるサイトのリストは、ユーザーが少なくとも読取りアクセス権を持っているサイトのリストになります。

9.3 SS_GET_SITE_PROPERTY

- **説明 :** サイト・プロパティを取得します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **property:** 取得するプロパティの名前（必須）。
- **戻り値 :**
  - **value:** リクエストされたサイト・プロパティの値。

  リクエストされたサイト・プロパティが存在しない場合、このパラメータは戻されませんが、サービス・コールは正常に完了します。
- **セキュリティ :** ユーザーは、サイトに対して少なくとも読取りアクセス権を持っている必要があります。このサービスは、Idoc スクリプトから実行できます。

9.4 SS_SET_SITE_PROPERTY

- **説明 :** サイト・プロパティを設定します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **property:** 設定または削除するプロパティの名前（必須）。
  - **value:** 属性の新しい値（オプション）。値が指定されない場合、プロパティは削除されます。
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、サイトに対して少なくとも書込みアクセス権を持っている必要があります。

9.5 SS_GET_NODE_PROPERTY

- **説明 :** ノード・プロパティを取得します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
  - **nodeId:** ノードの識別子（必須）
  - **property:** 設定または削除するプロパティの名前（必須）
- **戻り値 :**
  - **value:** リクエストされたノード・プロパティの値。リクエストされたプロパティが存在しない場合、このパラメータは戻されませんが、サービス・コールは正常に完了します。
- **セキュリティ :**
  - ユーザーは、ノードに対して少なくとも読取りアクセス権を持っている必要があります。
  - このサービスは、Idoc スクリプトから実行できます。

9.6 SS_SET_NODE_PROPERTY

- **説明 :** ノード・プロパティを設定します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **nodeId:** ノードの一意の識別子（必須）。
  - **property:** 設定または削除するプロパティの名前（必須）。
  - **value:** 属性の新しい値（オプション）。値が指定されない場合、プロパティは削除されます。
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、ノードに対して少なくとも書込みアクセス権を持っている必要があります。

9.7 SS_ADD_NODE

- **説明 :** 特定のサイト・ノードに子ノードを追加します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **nodeId:** ノードの一意の識別子（必須）。新規に追加されるノードは、このノードの子になります。
  - **newNodeId:** 新規子ノードの識別子（オプション）。この値が指定されない場合、新しく一意の識別子が生成されます。
  - **newNodeName:** 新規子ノードのラベル（オプション）。この値が指定されない場合、New Section というテキストが使用されます。
- **戻り値 :**
  - **newNodeId:** 新しく作成されたノードの識別子。
- **セキュリティ :** ユーザーは、新規子ノードを追加するノードに対して少なくとも書込みアクセス権を持っている必要があります。

9.8 SS_DELETE_NODE

- **説明 :** 特定のサイト・ノードとそのすべての子を削除します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
  - **nodeId:** 削除するノードの識別子（必須）
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、削除するノードに対して少なくとも削除アクセス権を持っている必要があります。

9.9 SS_MOVE_NODE

- **説明 :** 階層内でサイト・ノードを移動します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **nodeId:** 移動するノードの識別子（必須）。
  - **newParentId:** ノードの新しい親の識別子（必須）。
  - **insertAfterId:** 移動後のノードの前に位置する兄弟の識別子（オプション）。このパラメータが指定されない場合、ノードは *new_parent_id* の最初の子になります。
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、新規親ノードに対して少なくとも書込みアクセス権を持っている必要があります。

9.10 SS_GET_FIRST_NODE_ID

- **説明 :** サイトの最初のノードを取得します。サイト階層を列挙する場合に便利です。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
- **戻り値 :**
  - **firstId:** サイト階層における最初のノードの識別子。
- **セキュリティ :**
  - ユーザーは、ルート・ノードに対して少なくとも読取りアクセス権を持っている必要があります。
  - このサービスは、Idoc スクリプトから実行できます。

9.11 SS_GET_RELATIVE_NODE_ID

- **説明 :** 特定の相対ノード識別子を取得します。サイト階層を列挙する場合に便利です。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
  - **nodeId:** ノードの識別子（必須）
  - **relative:** parent、child、next または prior のいずれか（必須）
- **戻り値 :**
  - **relativeId:** サイト階層における相対ノードの識別子。
- **セキュリティ :**
  - ユーザーは、指定したノードに対して少なくとも読取りアクセス権を持っている必要があります。
  - このサービスは、Idoc スクリプトから実行できます。

9.12 SS_GET_SITE_AS_XML_EX2

- **説明 :** ノードおよび属性とともにサイト全体を XML として取得します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **includeProperties:** ノードのプロパティを結果セットとしてレスポンスに含めるかどうかを指定する True または False のブール型パラメータ（オプション）。
- **戻り値 :**
  - **siteXml:** サイトの XML 表現。
  - オプションで、XML ファイルに含まれるノードのプロパティを提供する 2 つの ResultSet（**StandardProperties** および **SiteStudioProperties**）が戻されます。nodeId パラメータにより、結果セットの各行が XML のノードに関連付けられます。
- **セキュリティ :** ユーザーは、階層のルート・ノードに対して少なくとも書込みアクセス権を持っている必要があります。戻される XML では、階層のノードがすべて列挙されます。この理由は、Site Studio では、変更可能なノードにユーザーがナビゲートできるようにツリー構造を表示する必要があるためです。

9.13 SS_CREATE_NEW_SITE_EX2

- **説明 :** コンテンツ・サーバーに空の新規サイトを作成します。
- **パラメータ :**
  - **siteId:** 作成するサイトの一意の識別子（必須）。識別子は、siteId のネーミング規則に準拠している必要があります。
  - **siteLabel:** サイトのラベル（オプション）。ラベルが指定されない場合、Unnamed Site というテキストが使用されます。
  - **siteType:** 作成するサイトのタイプ（asp または idoc）（オプション）。
  - **initialNodeId:** ルート・セクションの nodeId（オプション）。この値が指定されない場合、一意の識別子が作成されます。
  - **rootSectionActive:** ルート・セクションをアクティブとしてマークするかどうかを示します（オプション）。有効な値は、TRUE または FALSE です。
  - **rootSectionLabel:** ルート・セクションのラベル（オプション）。この値が指定されない場合、Home という値が割り当てられます。
  - **rootSectionUrlPageName:** ルート・セクションの urlPageName 値（オプション）。
  - **rootSectionUrlSecondaryPageName:** ルート・セクションの urlSecondaryPageName（オプション）。
  - **rootSectionUrlDirName:** ルート・セクションの urlDirName 値（オプション）。
- **戻り値 :**
  - **siteId:** 作成されたサイトの一意の識別子（必須）。
- **セキュリティ :** サービスを実行するユーザーは、Site Studio プロジェクト・ファイルをチェックインする際に使用されるセキュリティ・グループに対して書込みアクセス権を持っている必要があります（このファイルは、「Set Default Project Document Information」管理ページで構成されます）。

9.14 SS_GET_ALL_SITE_PROPERTIES

- **説明 :** サイト・プロパティの完全なセットを取得します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
- **戻り値 :** 値の行を 1 つ含む **SiteStudioProperties** という結果セットが戻されます。列名はプロパティ名です。
- **セキュリティ :**
  - ユーザーは、サイトに対して少なくとも読取りアクセス権を持っている必要があります。
  - このサービスは、Idoc スクリプトから実行できます。

9.15 SS_GET_ALL_NODE_PROPERTIES

- **説明 :** ノード・プロパティの完全なセットを取得します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
  - **nodeId:** ノードの識別子（必須）
- **戻り値 :**
  - 2 つの ResultSet（**StandardProperties** および **SiteStudioProperties**）が戻されます。各結果セットには 1 つの行が含まれます。列名はプロパティ名です。
  - StandardProperties 結果セットには、nodeId を示す 1 つの行および列のみが含まれます。
- **セキュリティ :**
  - ユーザーは、ノードのフォルダに対して少なくとも読取りアクセス権を持っている必要があります。
  - このサービスは、Idoc スクリプトから実行できます。

9.16 SS_CREATE_SITE_NAV_JS

- **説明 :** サイト・ナビゲーション・ファイルを作成します。これにより、sitenavigation.js、sitenavigationfunctions.js、sitenavigation.xml、sitenavigation.hda および sitenavigation_co.hda の各ファイルが再生成されます。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
- **戻り値 :**
  - **siteNavUrl:** サイト・ナビゲーション・ファイルの完全な HTTP URL
- **セキュリティ :** ユーザーは、ルート・ノードに対して少なくとも書込みアクセス権を持っている必要があります。

9.17 SS_GET_ALL_CUSTOM_NODE_PROP_DEFS

- **説明 :** 特定のサイトのカスタム・ノード・プロパティ定義を取得します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
- **戻り値 : CustomNodePropertyDefs** という ResultSet が戻されます。この結果セットには、**name**、**type** および **description** という列が含まれます。
- **セキュリティ :** ユーザーは、ルート・ノードに対して少なくとも書込みアクセス権を持っている必要があります。

9.18 SS_SET_ALL_CUSTOM_NODE_PROP_DEFS

- **説明 :** 特定のサイトのカスタム・ノード・プロパティ定義を設定します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。入力として **CustomNodePropertyDefs** という ResultSet を使用します。この結果セットには、**name**、**type** および **description** という列が含まれます（必須）。
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、ルート・ノードに対して少なくとも書込みアクセス権を持っている必要があります。

9.19 SS_GET_SITE_REPORT

- **説明 :** サイトで使用されているアイテムのレポートを取得します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、サイトに対して少なくとも書込みアクセス権を持っている必要があります。

9.20 SS_GET_SITE_PUBLISH_REPORT

- **説明 :** サイトで使用されているアイテムのレポートを取得します。これは、引用サイトの正確性を保証するために接続サーバーで使用するよう設計されています。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、サイトのルート・フォルダに対して少なくとも書込みアクセス権を持っている必要があります。

9.21 SS_EDIT_NATIVE_DOCUMENT

- **説明 :** Check Out and Open（COAO）の起動に使用できる HTML ページを戻します。このページには、COAO の起動に必要な JavaScript と <object> タグが含まれます。
- **パラメータ :**
  - **dDocName:** COAO 機能を使用して編集するファイルの dDocName（必須）
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。持っていない場合、COAO を使用することはできません。

9.22 SS_CLEAR_WEBSITE_ID

- **説明 :** 特定の dDocName の xWebsiteID フィールドを消去します。
- **パラメータ :**
  - **dDocName:** xWebsites フィールドを変更するコンテンツ ID（必須）
- **戻り値 :** なし。

9.23 SS_ADD_WEBSITE_ID

- **説明 :** 特定のドキュメントに対応する Web サイトのリストに特定の siteId を追加します。
- **パラメータ :**
  - **dDocName:** コンテンツ ID（必須）
  - **siteId:** サイトの一意の識別子（必須）
  - **fieldName:** xWebsites または xDontShowInListsForWebsite（必須）
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、特定のコンテンツ ID の最新のリビジョンに対して書込みアクセス権を持っている必要があります。

9.24 SS_REMOVE_WEBSITE_ID

- **説明 :** 特定のドキュメントに対応する Web サイトのリストから特定の siteId を削除します。
- **パラメータ :**
  - **dDocName:** コンテンツ ID（必須）
  - **siteId:** サイトの一意の識別子（必須）
  - **fieldName:** xWebsites または xDontShowInListsForWebsite（必須）
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、特定のコンテンツ ID の最新のリビジョンに対して書込みアクセス権を持っている必要があります。

9.25 SS_CHOOSE_WEBSITES

- **説明 :** リストから Web サイトを選択できる画面を表示します。これは、コンテンツ・サーバー UI の xWebsites フォーム要素のカスタマイズと組み合せて使用します。
- **パラメータ :**
  - **xWebsites:** Web サイト ID のカンマ区切りリスト。
- **戻り値 :** なし。
- **セキュリティ :** コンテンツ・サーバー内のセキュリティ・グループに対して読取りアクセス権を持っている任意のユーザーは、このサービスを起動できます。

9.26 SS_SWITCH_REGION_ASSOCIATION

- **説明 :** 特定のノードの特定のリージョンに関連付けられたコンテンツ ID を変更します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **nodeId:** ノードの識別子（必須）。
  - **property:** primaryUrl または secondaryUrl（必須）。他の値ではエラーが発生します。
  - **region:** リージョン識別子（必須）。
  - **value:** 設定するコンテンツ ID 値（必須）。
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、ノードに対して少なくとも書込みアクセス権を持っている必要があります。

9.27 SS_CLEAR_REGION_ASSOCIATIONS

- **説明 :** 切り替えられたリージョン関連付けの設定を消去します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **nodeId:** ノードの識別子（オプション）。このパラメータが指定されない場合、サイト全体の関連付けが消去されます。
  - **property:** primaryUrl または secondaryUrl（nodeId の指定時には必須）。
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、ノードに対して（nodeId を指定しない場合はサイトに対して）少なくとも書込みアクセス権を持っている必要があります。

9.28 SS_GET_SITE_DOMAINS

- **説明 :** サイトのドメイン・マッピング情報を戻します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
- **戻り値 :**
  - **ssSiteDomains:** サイトにマップされているドメインをリストした結果セット
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つのセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.29 SS_SET_SITE_DOMAINS

- **説明 :** サイトのドメイン・マッピング情報を設定します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
  - **ssSiteDomains:** サイトにマップするドメインをリストした結果セット
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.30 SS_SET_SITE_PROPERTIES

- **説明 :** 単一のサービス・コールで複数のサイト・プロパティを設定します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
  - **property:** 取得するプロパティの名前（必須）
  - **value:** リクエストされたサイト・プロパティの値
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.31 SS_GET_REGION_ASSOCIATIONS

- **説明 :** プライマリおよびセカンダリ URL のリージョン関連付けを戻します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
- **戻り値 :**
  - **primary:** サイトのすべてのプライマリ URL
  - **secondary:** サイトのすべてのセカンダリ URL
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.32 SS_GET_SITE_DEFINITION

- **説明 :** ユーザーが書込みアクセス権を持っているすべてのノードを示したサイト定義を XML で戻します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.33 SS_GET_SITE_DEFINITION_FOR_USER

- **説明 :** ユーザーが読取りアクセス権を持っているすべてのノードを示したサイト定義を XML で戻します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.34 SS_MAP_FRIENDLY_NAME

- **説明 :** dDocName 値をフレンドリ名に、またはフレンドリ名を dDocName にマップします。これは、SSUrlFieldName 構成設定の使用に関連します。
- **パラメータ :**
  - **inputName:** dDocName またはフレンドリ名の値（必須）
  - **direction:** fromDocName または toDocName（必須）
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.35 SS_GET_FRIENDLY_URL

- **説明 :** ドキュメントまたはセクションに対するフレンドリ URL を構成します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **ssDocName:** 表示するドキュメントの dDocName（オプション）。
  - **ssTargetNodeId:** ドキュメントを表示するセクションの識別子（オプション）。

    ssDocName または ssTargetNodeId のいずれかを指定する必要があります。
  - **ssTargetSiteId:** 指定された ssTargetNodeId を含むサイトの識別子。
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.36 SS_PARSE_FRIENDLY_URL

- **説明 :** 渡されたリンクのターゲットの siteId および siteRelativeUrl を戻します。
- **パラメータ :**
  - **ssFriendlyUrl:** サイトに存在するフレンドリ URL（必須）
- **戻り値 :**
  - **siteId:** サイトの一意の識別子
  - **siteRelativeUrl:** 入力されたフレンドリ URL の解析済 URL
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.37 SS_DECODE_LINK

- **説明 :** Site Studio リンクをデコードしてリンクが解決される場所を判別します。
- **パラメータ :**
  - **link:** サポートされる任意の形式による Site Studio リンク（必須）。
  - **siteId:** サイトの一意の識別子（必須）。
- **戻り値 :**（空白の可能性あり）
  - **targetIsSection:** True または False
  - **nodeLabel:** ノードのナビゲーション・ラベル。
  - **targetDocName:** リンクのターゲットとなる dDocName。
  - **targetNodeId:** リンクのターゲットとなるセクションの識別子。
  - **targetSiteId:** リンクのターゲットとなるサイトの識別子。
  - **targetIsSecondary:** True または False
  - **linkType:** 入力された Site Studio リンクのリンク・タイプ。
  - **errors:** 文字列値 unknown Error、noInputLink、invalidInputSiteId、invalidSiteId、invalidNodeId、invalidPathLink、noDocInfo、notSiteStudioUrl、badUrlFormat、cantFindUrlSection、parameterCount、parameterFormat のいずれかです。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.38 SS_BATCH_DECODE_LINK

- **説明 :** セット内にリストされたすべてのリンクおよびsiteIdに基づいてSS_DECODE_LINKを実行するために、SS_DECODE_LINKを対象にセットの受渡しと返却を行います。
- **パラメータ :** SS_DECODE_LINKにリスト入力するリンクおよびsiteIdのリストを格納したLINKというタイトルのセット。
- **戻り値 :** LINKSというタイトルの結果セット。入力行ごとに、SS_DECODE_LINKで解析された情報を含む行が1つ生成されます。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの1つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.39 SS_GET_LINK

- **説明 :** ssLinkのサーバー側リンクのサービス・バージョン。
- **パラメータ :**
  - **ssDocName:** 表示するドキュメントのdDocName（必須）
  - **ssTargetNodeId:** ドキュメントを表示するセクションの識別子（オプション）
  - **ssTargetSiteId:** 指定されたssTargetNodeIdを含むサイトの識別子（オプション）
  - **ssSourceNodeId:** ドキュメントのxWebsiteSection値を使用できない場合にドキュメントを表示するセクションの識別子（オプション）
  - **ssSourceSiteId:** 指定されたssSourceNodeIdを含むサイトの識別子（オプション）
- **戻り値 :** 結果は、ssLinkパラメータとして戻されます。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの1つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.40 SS_GET_NODE_LINK

- **説明 :** ssNodeLinkのサーバー側リンクのサービス・バージョン。
- **パラメータ :**
  - **ssTargetNodeId:** ドキュメントを表示するセクションの識別子（必須）
  - **ssTargetSiteId:** 指定されたssTargetNodeIdを含むサイトの識別子（オプション）
  - **ssSourceNodeId:** ドキュメントのxWebsiteSection値を使用できない場合にドキュメントを表示するセクションの識別子（オプション）
  - **ssSourceSiteId:** 指定されたssSourceNodeIdを含むサイトの識別子（オプション）
- **戻り値 :** 結果は、ssNodeLinkパラメータとして戻されます。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの1つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.41 SS_GET_LINK_MANAGEMENT_REPORT

- **説明 :** Site Studio サイト内のリンクに関する情報を取得します。
- **パラメータ :** なし。
- **戻り値 :** 次の列を含む Manifest という結果セット。
  - **siteId:** サイトの一意の識別子（必須）。
  - **layoutResultSet:** 次の値が含まれます。
    - **nodeId:** セクションの識別子。
    - **dDocName:** ドキュメントの名前。
    - **isPrimaryUrl:** 結果がプライマリ・ページとして指定されているかどうか。
  - UrlDataFiles: 次の値が含まれます。
    - **nodeId:** セクションの識別子。
    - **dDocName:** ドキュメントの名前。
    - **isPrimaryUrl:** 結果がプライマリ・ページとして指定されているかどうか。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.42 SS_GET_ENVIRONMENT_PROPERTY_NAMES

- **説明 :** 環境プロパティとして識別されるこのサイトのプロパティを取得します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
- **戻り値 :** EnvironmentProperties という結果セットが戻されます。この結果セットには、環境プロパティとして識別されたプロパティごとに 1 行を持つ name という単一の列が含まれます。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.43 SS_SET_ENVIRONMENT_PROPERTY_NAMES

- **説明 :** 指定されたサイトに対して定義済の環境プロパティを設定します。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - EnvironmentProperties という結果セット。この結果セットには、環境プロパティとして識別されるプロパティごとに 1 行を持つ name という単一の列が含まれます。
- **戻り値 :** なし。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.44 SS_GET_WEBLAYOUT_URL

- **説明 :** ssWeblayoutUrl というラベルのパラメータで完全な Web レイアウト URL を戻します。
- **パラメータ :**
  - **ssWebLayoutParam:** dDocName、またはパス groups/ で始まる Web レイアウト URL（必須）。
- **戻り値 :** ssWeblayoutUrl というラベルのパラメータに含まれた完全な Web レイアウト URL。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

9.45 SS_GET_SEARCH_RESULTS

- **説明 :** これは、通常の CS GET_SEARCH_RESULTS サービスのラッパーです。これにより、いくつかの Site Studio 機能をフラグで簡単に指定し、実際の問合せ構文をサーバー上で構成できます。また、標準の SearchResults 結果セットを変更して、すべての検索結果について Site Studio で使用しやすい URL を格納した ssUrl という追加列を含めることができます。
- **パラメータ :**
  - **siteId:** サイトの一意の識別子（必須）。
  - **ssLimitScope:** サイト内のみに有効範囲を制限するかどうかを示す True または False の値（オプション）。
  - **ssUserSearchText:** ユーザー指定の検索テキスト（オプション）。
  - **ssWebsiteObjectType:** オブジェクト・タイプ（オプション）。
  - **computeFriendlyUrls:** ssUrl を含むように結果を変更するかどうかを示す True または False の値（オプション）。
- **戻り値 :** Search Results という名前の結果セット。
- **セキュリティ :** ユーザーは、コンテンツ・サーバーの 1 つ以上のセキュリティ・グループに対して書込みアクセス権を持っている必要があります。

10

JSP によるサーバー側スクリプト機能

この章の内容は次のとおりです。

10.1 JSP によるサーバー側スクリプト機能について

Content Server により提供される Web ページ用のネイティブのサーバー側スクリプト言語は、Idoc スクリプトです。そのため、Idoc スクリプトは、スクリプト言語としてデザイナのほとんどのサンプル・フラグメントに選択されており、説明を簡略化する目的でこのドキュメントの一部でも使用されています。

Idoc スクリプトに加え、Web ページではサーバー側言語として JavaServer Pages（JSP）を使用できます。JSP 対応の Web サイトの場合、レイアウト・ページやフラグメントの構成に際して異なるアプローチが必要です。

- 10-2 ページの「埋込み JSP ページ」
- 10-6 ページの「JSP フラグメント」

10.2 埋込み JSP ページ

レイアウト・ページは、コンテンツ・サーバーから提供される JSP ページとして実装できます。このモデルでは、埋込み Tomcat サーブレット・エンジンを使用してサーバー側 JSP を実行し、完成した Web ページを配信します。埋込みエンジンは、コンテンツ・サーバー環境と緊密に統合されているため、JSP コンテキスト内で Idoc 環境の各機能を共有できます。この共有環境により、複数の処理が簡略化されます。

動的ページは、JSP として識別されると、評価のために Tomcat エンジンに渡されます。この受渡しには、コンテンツ・サーバー・フレームワーク環境への参照が含まれます。埋込みの idcServer.ServerBean クラスの使用により、JSP ページでは Idoc 変数を取得および設定し、サービスを実行できます。この結果、JSP ページで Idoc スクリプトを実行できます。

JSP ページは、次の領域で HCSP ページと大きく異なります。

- 10-2 ページの「JSP の ss_layout_head_info リソース・インクルード」
- 10-5 ページの「JSP の一般的なマークアップの違い」

10.2.1 JSP の ss_layout_head_info リソース・インクルード

Idoc スクリプトの ss_layout_head_info リソース・インクルードの詳細は、2-12 ページの「ss_layout_head_info リソース・インクルード」を参照してください。JSP の ss_layout_head_info リソース・インクルードでは、レイアウト・ページのマークアップが多少異なります。

```
<%@include file="/WEB-INF/ss_layout_head_info.jsp"%>
```

インクルードは、レイアウト・ページ内で可能なかぎり早く指定します。デフォルトでは、<head> タグに配置されます。

ss_layout_head_info ファイルのコンテンツにより、ページ・リクエストに対する標準の JSP コンテキストが設定されます。その後、serverbean オブジェクトの使用により、コードで次のように標準の Idoc ss_layout_head_info リソース・インクルードが実行されます。

- 定式名 serverbean を使用して idcserver.ServerBean のインスタンスを作成します。

  ```
  <%@ page import = "idcserver.*" %>
  <jsp:useBean id="serverbean" class="idcserver.ServerBean"/>
  ```

- sitenavigation.js に実装された JavaScript オブジェクト階層に似た動作をサポートする別のヘルパー・オブジェクト ss_navbean を作成します。

  ```
  <%@ page import = "sitestudio.SSNavigationNode" %>
  <jsp:useBean id="ss_navbean" class="sitestudio.SSNavigationBean"/>
  ```

- serverbean および ss_navbean オブジェクトを初期化し、一般的に使用される値を Idoc 環境から取得して JSP の request オブジェクトに属性として格納します。これにより、各値をこのページの JSP フラグメント・インスタンスで使用できます。

```
<%
     serverbean.init(request);

     String ssSiteId = serverbean.evalIdcScp("siteId");
     request.setAttribute("ssSiteId", ssSiteId);

     String ssUrlPrefix = serverbean.evalIdcScp("ssUrlPrefix");
     request.setAttribute("ssUrlPrefix", ssUrlPrefix);

     String ssNodeId = serverbean.evalIdcScp("nodeId");
     request.setAttribute("ssNodeId", ssNodeId);

     ss_navbean.init(ssSiteId, ssUrlPrefix);
```

- 標準の ssNavNodeList 結果セットを処理して、定式配列 ss_navpath を作成します。（この配列は、JavaScript の g_navNode_Path 配列の JSP バージョンです。）

```
ServerResultSet rs = serverbean.getResultSet("ssNavNodeList");
String ss_navpath[] = new String[rs.getNumRows()];
     {
          rs.first();
          int i = 0;
          while (rs.isRowPresent())
          {
               ss_navpath[i++] = rs.getStringValue("nodeId");
               rs.next();
          }
     }
```

- 定式変数 ss_navroot を作成します。（この変数は、g_navNode_Root の JSP バージョンです。）

```
SSNavigationNode ss_navroot = ss_navbean.getRootNode();
```

- これらの変数に対する参照を JSP の request オブジェクトに格納し、このページの JSP フラグメント・インスタンスで使用できるようにします。

```
request.setAttribute("ss_serverbean", serverbean);
request.setAttribute("ss_navbean", ss_navbean);
request.setAttribute("ss_navpath", ss_navpath);
request.setAttribute("ss_navroot", ss_navroot);
```

- より一般的に使用される値を Idoc 環境から取得して JSP の request オブジェクトに属性として格納します。これにより、各値をこのページの JSP フラグメント・インスタンスで使用できます。

```
String ssServerRelativeSiteRoot =
serverbean.evalIdcScp("ssServerRelativeSiteRoot");
request.setAttribute("ssServerRelativeSiteRoot", ssServerRelativeSiteRoot);

String ssSiteRelativeUrl = serverbean.evalIdcScp("siteRelativeUrl");
request.setAttribute("ssSiteRelativeUrl", ssSiteRelativeUrl);

String HttpCgiPath = serverbean.evalIdcScp("HttpCgiPath");
request.setAttribute("HttpCgiPath", HttpCgiPath);

String HttpRelativeWebRoot = serverbean.evalIdcScp("HttpRelativeWebRoot");
request.setAttribute("HttpRelativeWebRoot", HttpRelativeWebRoot);
```

```
String HttpRelativeFragmentsRoot =
serverbean.evalIdcScp("HttpRelativeFragmentsRoot");
request.setAttribute("HttpRelativeFragmentsRoot", HttpRelativeFragmentsRoot);
```

- Cache-Control ヘッダー値を評価して JSP の response オブジェクトに設定します。

```
serverbean.evalResInc("ss_compute_cache_control_header");
String ssCacheControl = serverbean.evalIdcScp("ssCacheControl");
if ssCacheControl != null ) && ( ssCacheControl.length() != 0 )
   {
      response.setHeader( "Cache-Control", ssCacheControl );
   }
```

- SS_FRIENDLY_URL ヘッダー値を評価して JSP の response オブジェクトに設定します。(値は SSPU で使用されます。)

```
String ssFriendlyUrl = serverbean.evalIdcScp("#active.SS_FRIENDLY_URL");
if  ssFriendlyUrl != null && ssFriendlyUrl.length() != 0 )
   {
       response.setHeader( "SS_FRIENDLY_URL", ssFriendlyUrl );
   }
```

- 必要に応じて JSP セッション Cookie を初期化します。

```
if (session.isNew())
    {
        Cookie mySessionCookie = new Cookie("JSESSIONID", session.getId());
        mySessionCookie.setPath("/");
        response.addCookie(mySessionCookie);
    }
```

- ヘルパー・クラスおよびインスタンスで ssJSP.link および ssJSP.nodeLink サーバー側ナビゲーション・メソッドを実装します。

```
class ssJSPUtils
{
   idcserver.ServerBean m_bean = null;

   public ssJSPUtils() {};
   public void init (idcserver.ServerBean theBean) { m_bean = theBean; }

   public String link(String docId) { return link (docId, "", ""); }
   public String link(String docId, String nodeId) { return link (docId, nodeId,
"", ""); }
   public String link(String docId, String nodeId, String siteId)
    {
     String retval = "";
     try
      {
       retval = m_bean.evalIdcScp("ssLink('" + docId + "', '" + nodeId + "', '" +
siteId +  "')");
      }
     catch (Exception e)
      {}
     return retval;
  }
   public String nodeLink(String nodeId) { return nodeLink (nodeId, ""); }
   public String nodeLink(String nodeId, String siteId)
    {
       String retval = "";
       try
        {
           retval = m_bean.evalIdcScp("ssNodeLink('" + nodeId + "', '" + siteId +
"')");
        }
```

```
            catch (Exception e)
            {}
          return retval;
        }
    }
    ssJSPUtils ssJSP = new ssJSPUtils();
    ssJSP.init(serverbean);
    %>
```

- 最後に、serverbean オブジェクトを使用して標準の Idoc ss_layout_head_info リソースを組み込みます。

```
<%=serverbean.evalResInc("ss_layout_head_info")%>
```

10.2.2 JSP の一般的なマークアップの違い

serverbean オブジェクトの使用を通じて、JSP では Idoc スクリプトを正常に実行できます。つまり、使用方法を大きく変更することなく、Idoc 環境で JSP をサポートできます。<ssinfo> XML データ・アイランド、リージョン、要素およびフラグメントのメカニズムは、HCSP ページとほぼ同じ方法で構成されます。ただし、Idoc の組込みに使用されるマークアップには、大きな違いがあります。

この方法を使用して、次のコードを置き換えます。

```
<!-- SS_BEGIN_OPENREGIONMARKER(region1)-->
<!--$SS_REGIONID="region1"-->
<!--$include ss_open_region_definition -->
<!-- SS_END_OPENREGIONMARKER(region1)-->
```

前述のコードは、次のコードで置き換えられます。

```
<!-- SS_BEGIN_OPENREGIONMARKER(region1)-->
<%=serverbean.evalIdcScp("SS_REGIONID=\"region1\"")%>
<%=serverbean.evalResInc("ss_open_region_definition")%>
<!-- SS_END_OPENREGIONMARKER(region1)-->
```

元のコードの最初と最後の行は、Site Studio デザイナで使用される単純なコメント・マーカーです。2 行目では、値を宣言し、3 つの変数に割り当てています。3 行目では、Idoc リソース・インクルードをインクルード（および実行）しています。

スクリプトは、serverbean.evalIDcScp() コールにラップされることで実行されます。リソース・インクルードは、serverbean.evalResInc() コールにより実行されます。SS_REGIONID= value には、追加の \" エスケープ処理が必要です。また、いくつかのマークアップでは、さらに別のエスケープ処理が必要です。特に、<ssinfo> XML データ・アイランドのパラメータは、データを有効な XML として解析できるように、CDATA セクション内に組み込む必要があります。

次のコードを置き換えます。

```
<region id="region1" name="region1" flags="111000000" metadata=""
dccommand="ssIncInlineDynamicConversion(SS_DATAFILE)">
     <!--$region1_ACTIONS="E",region1_DCCOMMAND="ssIncInlineDynamicConversion
(SS_DATAFILE)" -->

     <switchregioncontent createnewxml="false" createnewnative="false"
choosemanaged="true" chooselocal="false" choosenone="false">
<choosemanagedquerytext corecontentonly="FALSE">        <![CDATA[xWebsiteObjectType
<Matches> `Data File` <OR> xWebsiteObjectType <Matches> `Native Document`]]>
     </choosemanagedquerytext>
     </switchregioncontent>
     </region>
```

前述のコードは、次のコードで置き換えられます。

```
<region id="region1" name="region1" flags="111000000" metadata=""
dccommand="ssIncInlineDynamicConversion(SS_DATAFILE)">
          <![CDATA[<%=serverbean.evalIdcScp("region1_ACTIONS=\"E\",
region1_DCCOMMAND=\"ssIncInlineDynamicConversion(SS_DATAFILE)\"") %>]]>

     <switchregioncontent createnewxml="false" createnewnative="false"
choosemanaged="true" chooselocal="false" choosenone="false">
<choosemanagedquerytext corecontentonly="FALSE">        <![CDATA[xWebsiteObjectType
<Matches> `Data File` <OR> xWebsiteObjectType <Matches> `Native Document`]]>
     </choosemanagedquerytext>
     </switchregioncontent>
     </region>
```

Site Studio コンポーネントの構造に影響する変更はありません。マークアップの違いにかかわらず（この違いはデザイナにより管理されます）、リージョン、要素およびフラグメントの操作方法は同じです。

serverbean と統合環境を使用することで、標準の Idoc スクリプトおよび JavaScript フラグメントは、元のコードを変更することなくすべて JSP ページで使用できます。コントリビューション・リージョンの動作も同じであり、実装によって標準のコンポーネント・リソース・インクルードが組み込まれます。

10.3 JSP フラグメント

JSP 環境では、JSP でコーディングされたフラグメントを Site Studio が提供します。JSP フラグメントは、ロジックの大部分が Java コードで記述されているか、サーバー側で実行できる JSP タグを含んでいます。このため、追加の規則が必要となりますが、その一部は ss_layout_head_info.jsp で確認できます（10-2 ページの「JSP の ss_layout_head_info リソース・インクルード」を参照）。

一部の定式オブジェクトおよびデータがページの初めに作成され、JSP の request オブジェクトにおけるフラグメント・コードで使用できるようになります。ただし、依然として JSP コードをページ実行環境に組み込む必要があります。ssIncludeXml() メカニズムは、静的テキストや Idoc スクリプトを組み込む場合には問題ありませんが、JSP のインクルードや実行には使用できません。コードは、ページにインライン配置するか、正式な JSP インクルード・メカニズムを使用してインクルードする必要があります。

JSP では、2 つのインクルード・メカニズムを使用できます。1 つ目のメカニズムは、<%@include...%> 構文を含んでおり、ss_layout_head_info.jsp リソース・インクルードに使用されます（10-2 ページの「JSP の ss_layout_head_info リソース・インクルード」を参照）。2 つ目のメカニズム（JSP のインクルードに推奨されるメカニズム）は、組込みの <jsp:include> タグです。このメカニズムでは、フラグメント・コードの実行コンテキストがレイアウト・ページから分離されるため、フラグメント・コードがレイアウト・ページの保持する変数を破損することがありません。

Site Studio に付属するフラグメントの 1 つに、JSP Nav Plain Horizontal という名前のものがあります（デザイナ・ガイドのサンプル・フラグメントに関する項を参照）。このフラグメントでは、JSP コードが実際にフラグメント・アセットとして格納され、そのアセットがフラグメント内のインライン・スニペットから参照されていることを確認できます。

フラグメント・スニペットは、次のようになります。

```
<!-- SS_BEGIN_SNIPPET(fragment2,2)-->
<%=serverbean.evalIdcScp("ssFragmentInstanceId=\"fragment2\"")%>
<jsp:include
 page="/WEB-INF/fragments/jspnavplainhorizontal/JSPNavPlainHorizontal.jsp">
</jsp:include>
<!-- SS_END_SNIPPET(fragment2,2)-->
```

SS_BEGIN_SNIPPET および SS_END_SNIPPET の行は、デザイナに使用される単純なコメントです。ssFragmentInstanceId の評価は、JSP 用にマークアップされた標準の処理です。

特別な部分は、インラインで直接インクルードされているスニペット・テキストです。このテキストは、フラグメント実装の本体を組み込む jsp:include タグです。JSP ファイル（JSPNavPlainHorizontal.jsp）は、JavaScript ファイル、CSS ファイルまたはグラフィックと同じように、フラグメント・ライブラリにアセットとして格納されています。フラグメント・ライブラリがコンテンツ・サーバーに追加されると、これらのアセットは、実行時によりアクセスしやすい場所に抽出されます（10-8 ページの「JSP フラグメント・アセットのデプロイメント」を参照）。

次に、JSPNavPlainHorizontal.jsp の詳細を示します。

標準のクラス・インポート：

```
<%@ page import = "sitestudio.SSNavigationBean" %>
<%@ page import = "sitestudio.SSNavigationNode" %>
<%@ page import = "java.util.*" %>
```

request オブジェクトからグローバル・ヘルパー・オブジェクトを取得：

```
<%
idcserver.ServerBean serverbean =
(idcserver.ServerBean) request.getAttribute("ss_serverbean");

sitestudio.SSNavigationBean ss_navbean =
(sitestudio.SSNavigationBean) request.getAttribute("ss_navbean");

sitestudio.SSNavigationNode ss_navroot =
(sitestudio.SSNavigationNode) request.getAttribute("ss_navroot");

final String ss_navpath[] = (String [])request.getAttribute("ss_navpath");
```

Idoc 実行コンテキストからフラグメント・パラメータ値を取得：

```
String ssTextColor  = serverbean.evalIdcScp("getValue(\"#active\",
                        ssFragmentInstanceId & \"_ssTextColor\")");
String ssHoverColor = serverbean.evalIdcScp("getValue(\"#active\",
                        ssFragmentInstanceId & \"_ssHoverColor\")");
String ssFocusColor = serverbean.evalIdcScp("getValue(\"#active\",
                        ssFragmentInstanceId & \"_ssFocusColor\")");
String ssClassName  = serverbean.evalIdcScp("getValue(\"#active\",
                        ssFragmentInstanceId & \"_ssClassName\")");
String ssShowHome   = serverbean.evalIdcScp("getValue(\"#active\",
                        ssFragmentInstanceId & \"_ssShowHome\")");
String ssSeparator  = serverbean.evalIdcScp("getValue(\"#active\",
                        ssFragmentInstanceId & \"_ssSeparator\")");
```

フラグメント実装コード

```
////////////////////////////////////////////////////////////////////////////
// Class : JSPNavPlainHorizontal
// Comments :
////////////////////////////////////////////////////////////////////////////
```

（実装の詳細は省略。）

パラメータを使用して前述のコードのインスタンスを起動し、Display メソッドを実行するコード：

```
JSPNavPlainHorizontal x =
  new JSPNavPlainHorizontal(ssTextColor, ssHoverColor, ssFocusColor,
  ssSeparator, ssClassName, ssShowHome);
x.Display(pageContext.getOut(), ss_navroot);
%>
```

これで完了です。このフラグメント・コードがインクルードおよび実行されると、結果がページ・コンテキストに順次戻されて、ブラウザに配信されます。

10.3.1 JSP フラグメント・アセットのデプロイメント

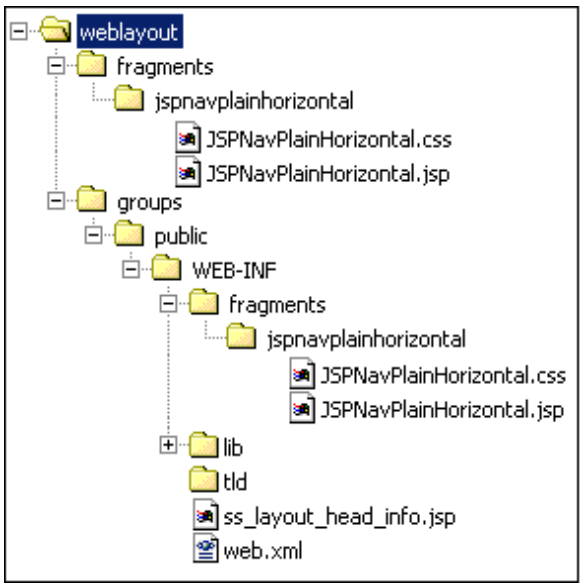
フラグメント・アセットは、JavaScript ファイル、CSS ファイルおよびグラフィックと同様に、すべて weblayout/fragments ディレクトリに抽出されます。このディレクトリのフラグメント・アセットは、実行時にクライアント側コードにより選択されます。

一方で、JSP アセットは、JSP ページの実行エンジンが選択できるようにデプロイする必要があります。Tomcat（および一般的な JSP）では、アプリケーション・コンテキストを厳密に区分する必要があります。Content Server では、特定のセキュリティ・グループが JSP 対応であると宣言することで、コンテキストを実装します。この効果により、これらのいずれかのグループから提供される JSP ページでアクセスできるのは、サイト階層のコンテキスト・ルート以下のファイルのみとなります。結果として、JSP ページは weblayout/fragments ディレクトリにアクセスできなくなるため、JSP 対応の各セキュリティ・グループの制御下に WEB-INF という追加ディレクトリが作成されます。フラグメント・ライブラリは、アップロード（または編集）されると、weblayout/fragments および weblayout/groups/<group name>/WEB-INF/fragments に抽出されます。

これにより、複数のセキュリティ・グループが有効化され、各グループのページは単一の管理対象フラグメント・ライブラリのアセットにアクセスできます。また、JSP 標準タグ・ライブラリ（JSTL）仕様を実装するファイルのデプロイに使用される WEB-INF/lib ディレクトリもあります。

次の図は、2 つの場所にデプロイされた JSP Nav Plain Horizontal フラグメントのアセットを示しています（他のフラグメントとセキュリティ・グループは、この図には含まれていません）。

図 10-1 JSP Nav Plain Horizontal フラグメントのコンテンツ



複数の場所にデプロイされるフラグメントのアセットでは、この構造内で冗長な部分が存在します。CSS および JavaScript ファイルは、WEB-INF ディレクトリに配置されますが、通常はその場所で使用されることはありません。JSP ファイルも、weblayout/fragments ディレクトリに配置されますが、使用されることはありません。前述のとおり、これらすべての処理は、JSP フラグメントと様々なセキュリティ・グループに対応するために実行されます。

10.3.2 JSP のコーディング技術

Site Studio に付属するすべてのサンプル JSP フラグメントでは、実装に JSP スクリプト要素を使用します。スクリプトレット（<%...%>）および式（<%=...%>）では、ナビゲーション構造にアクセスするために Bean によってサポートされる Java コードを使用します。

（request の）ss_navbean オブジェクトは、sitestudio.SSNavigationBean クラスのインスタンスです。このオブジェクトは、ss_layout_head_info.jsp によりインスタンス化および初期化されます。

このオブジェクトは、JSP フラグメントに役立つ 2 つの public メソッドをサポートします。

```
public SSNavigationNode getRootNode()
```

これにより、Web サイトのルート・ノードに対応する SS_NavigationNode オブジェクトが戻されます。

```
public Document getDocument()
```

これにより、sitenavigation.xml のコンテンツとともに事前ロードされた XML DOM オブジェクトが戻されます。

（request の）ss_navroot オブジェクトは、sitestudio.SSNavigationNode クラスのインスタンスです。これは、ss_navbean.getRootNode() から戻されるものと同じオブジェクトです。（この機能は、JavaScript の g_navNode_Root オブジェクトの機能と同じです。）

ss_navpath 配列は、機能的に JavaScript の g_navNode_Path 配列と同じノード（セクション）識別子の文字列配列です。

sitestudio.SSNavigationNode クラスでは、ナビゲーション階層をナビゲートするのに使用できる public メソッドがサポートされます。

```
public SSNavigationNode getParent  ()
public int getLevel ()
public String getLabel ()
public String getRawLabel ()
public String getId ()
public String getHref ()
public Vector getSubNodes ()
public NameNodeMap getAttMap
```

SS_Fragments_JSP に含まれる JSP Test Fragment には、使用可能な JSP 実装に関する他のいくつかの概要コードが含まれます。

Idoc サーバー側リンク関数の <!--$link(...)--> および <!--$nodeLink(...) は、JSP の <%=ssJSP.link(...)%> および <%=ssJSP.nodeLink(...)%> により置き換えられます（詳細は、10-2 ページの「JSP の ss_layout_head_info リソース・インクルード」を参照してください）。

2-8 ページの「サーバー側スクリプトのハイパーリンク」も参照してください。

11

ASP によるサーバー側スクリプト機能

この章の内容は次のとおりです。

11.1 ASP によるサーバー側スクリプト機能について

ASP をレイアウト・ページ用のスクリプト言語として選択すると、ページはコンテンツ・サーバーに格納され、そこで管理されます。ただし、ASP スクリプトは、コンテンツ・サーバーでは実行されません。かわりに、コンテンツ・サーバーでは、ASP ファイルを処理できないことを認識し、処理の可能な Web サーバーにそれらのファイルを渡します。

ASP ファイルは、コンテンツ・サーバーで処理されないため、Idoc スクリプトを含むことはできません（JSP の動作とは異なります）。また、コンテンツ・サーバーでページが処理されないことから、ASP で記述されたレイアウト・ページを Site Studio で使用するには、多くの代替インフラストラクチャが必要です。

大きな違いの 1 つは、ASP のレイアウト・ページでコンテンツ・サーバー関連の情報が必要とされる場合、ページを実行中の ASP 環境はコンテンツ・サーバー環境から完全に分離されているため、情報を取得するためにコンテンツ・サーバーにサービス・リクエストを戻す必要があるということです。これを行うには、SOAP リクエストを作成し、IdcCommandX ActiveX コントロールを使用してそのリクエストをコンテンツ・サーバーに送信します。

リリース 7.5 より前の Content Server では、Site Studio ASP サイトを動作させるために SOAP コンポーネントをインストールする必要があります。（Content Server 7.5 には、SOAP 機能が組み込まれているため、追加のコンポーネントは不要です。）また、すべてのリリースでコンテンツ・サーバーに IdcCommandX.ocx をインストールする必要があります。詳細は、インストレーション・ガイドの「コンテンツ・サーバーでの Active Server Pages の構成」を参照してください。

Site Studio では、ASP を含むすべてのサイト・タイプにフレンドリ URL を使用します。つまり、URL は次のようには表示されません。

```
http://<domain>/stellent/websites/get_page.asp?nodeId=42
```

かわりに、フレンドリ URL では次のようにより自然に表示されます。

```
http://<domain>/mysite/index.htm
```

この新しいスタイルの URL でも、最終的には ASP ページを識別する必要があります。このマッピングは、新しい Site Studio URL マッピング・プラグインで実行されます。これは、わかりやすい表示の URL を取得して、古いスタイルの URL に似た get_page.asp URL にマップする Web サーバー・フィルタのプラグインです。このプラグインのメリットは、get_page.asp URL をエンド・ユーザーから完全に隠蔽できることです。また、? を含まない新しい URL は、Google などの検索エンジンによるクロール対象としてより適しています。

11.2 ASP サポート・ファイル

ASP サイトには、次の 6 つの追加サポート・ファイルが必要です。

- 11-3 ページの「get_page.asp」
- 11-4 ページの「asp_includes.asp」
- 11-4 ページの「ss_layout_head_info.asp」
- 11-4 ページの「asp_link_functions.asp」
- 11-4 ページの「asp_datafile_utils.asp」
- 11-4 ページの「global.asa」

これらのファイルは、Site Studio のインストール後にコンテンツ・サーバーの weblayout 領域の websites フォルダで確認できます。

後続の詳細情報は、わかりやすい表示の URL が get_page.asp スタイルの URL にマップされてから発生する処理に関連しています。このドキュメントでは、URL が実際にどのように一方の形式から他方の形式にマップされるかについては説明しません（単に、マッピング表が <install_dir>/<instance_name>/data/sitestudio/ss_urlmap.hda ファイルに格納され、Web サーバー・フィルタで自動的に使用されることのみを示します）。Web サイト・アドレスの変更は、自動的に Web サーバー・フィルタに伝播されるため、このファイルを手動で操作する必要はありません。

11.2.1 get_page.asp

これは、ASP サイト内のすべてのリンクの主要なエントリ・ポイントです。すべてのフレンドリ URL と、すべての javascript:link() および javascript:nodelink() 関数は、URL 問合せ文字列で指定された追加パラメータとともに、最終的には get_page.asp の URL になります。

たとえば、次の関数があるとします。

```
Javascript:nodelink('234')
```

これは次のように変換されます。

```
http://<server_name>/<install_dir>/websites/get_page.asp?nodeId=234
```

このファイルの先頭には、ASP エンジンによる ASP ページの配信方法を制御するいくつかの重要なエントリがあります。

```
<%@ Language=VBScript %>
```

この行は、ASP エンジンに対して、ページで使用されるスクリプト言語が VBScript であることを示しています。（これは、ファイルの 1 行目に配置する必要があります。）

```
<%Response.CacheControl="No-Cache"%>
<%Response.Expires=-1%>
```

これらの行は、ブラウザに対して、戻されたページをキャッシュしないように指示しています。ページがブラウザにキャッシュされると、ページに対する変更が反映されません。ブラウザは、Web サーバーにページの再生成を依頼せずに、ページをキャッシュからリロードするためです。（変更のないページであることがわかっている場合は、このフラグをオフにできます。結果として、パフォーマンスが向上します。）

Site Studio には、Site Studio の動的 Web サイトのパフォーマンスを大幅に向上するように設定できるリバース・キャッシュ・プロキシ・サーバーの使用をサポートするいくつかの機能が含まれます。（多くの場合、このようなプロキシ・サーバーの使用を通じて実現されるパフォーマンスの向上により、動的サイトを静的サイトに公開する必要がなくなります。）

リバース・プロキシを使用する場合、この No-Cache の値をオフにして、より適切な値に設定できます。これを行うには、ノード・プロパティの Max Age を使用します。ノードにこの値を指定する場合、No-Cache の値を変更して任意の最大有効期限を指定します。たとえば、次のようになります。

```
<%Response.CacheControl="max-age=200, s-maxage=200"%>
<%Response.Buffer=true%>
```

これらの行は、ASP エンジンに対して、ページ全体の準備が完了するまでレスポンスを送信しないように指示しています。この指定がないと、ページは多くの小さな要素に分割されて送信されるため、ページ全体を一度に送信するよりも効率が悪くなります。

get_page.asp ファイルの残り部分は、SS_GET_PAGE サービス機能と同じです。必要に応じてユーザーのログインをチェックしてから、（SOAP と IdcCommandX を使用して）コンテンツ・サーバーにコールを戻し、リクエストされたページを編成するのに必要なすべての情報を取得します。

コンテンツ・サーバーに対するコールをできるかぎり少なくするため、ページの様々な要素を編成するのに必要なすべての情報が一度に収集されます。その後、すべての情報は、レイアウト・ページを構成する様々なリージョンやフラグメントなどで使用できるように、単一ページの編成中に ASP の Session オブジェクトに格納されます。（get_page.asp、リージョン、フラグメントなどについては、後で詳しく説明します。）

11.2.2 asp_includes.asp

このファイルには、Site Studio ASP サイトの運用に使用される実際の VBScript コードのほとんどが含まれます。また、コンテンツ・サーバーに対してサービス・コールを戻すための SOAP リクエストを作成および実行するすべてのコードが含まれます。

このファイルには、開始および終了リージョン・マーカーを実装するすべてのコードも含まれます。

11.2.3 ss_layout_head_info.asp

このファイルは、HCSP サイトにおける同様の名前の標準 Idoc インクルードの ASP バージョンです。このファイルは、すべてのレイアウト・ページの先頭でインクルードされ、sitenavigation.js、sitenavigationfunctions.js および contributor.js の追加インクルードを格納しています。（これらの .js ファイルは、すべての Web サイトに対してコンテンツ・サーバーにより生成され、コントリビュータ・アプリケーションをサポートするサイト階層やコードに関する情報を含んでいます。）

その他にも、レイアウト・ページがコントリビューション・モードに移行すると（通常は [Ctrl]+[Shift]+[F5] を押して移行）、このファイルにより（コントリビュータ・アプリケーション用の）OBJECT タグがレイアウト・ページにインクルードされます。

11.2.4 asp_link_functions.asp

このファイルには、サーバー側リンク関数の ssLink および ssNodeLink の実装が含まれます。これらは、現在の Site Studio の主要な要素である新規フレンドリ URL およびリンク・オプションの一部です。ssLink では、評価のためにコンテンツ・サーバーへのラウンドトリップが必要ですが、ssNodeLink は、現在のサイト内のノードを対象とするかぎり、サーバーに戻すことなく評価できます。

11.2.5 asp_datafile_utils.asp

このファイルには、サーバー側リンク（ssLink および ssNodeLink）を処理するためのサポート関数と、データ・ファイルで使用できる選択された Idoc タイプのトークンが含まれます。一般的に、ASP サイトでは、データ・ファイルで Idoc がサポートされないことに注意してください。サポートされるリンクと新規トークンの詳細は、11-11 ページの「ASP リンク関数」および 11-12 ページの「ASP データ・ファイル」を参照してください。

11.2.6 global.asa

これは、ASP アプリケーション用のグローバル変数ファイルです。Site Studio の APS サポートでは、このファイルを使用して、ASP アプリケーションの起動時に ASP の Application オブジェクトに追加される 2 つの情報を格納します。2 つの値（ssIdcReference および ssIdcWebRoot）は、Site Studio の ASP レイアウトがコンテンツ・サーバーに SOAP サービス・リクエストを戻す際に、通信先のコンテンツ・サーバーの場所を特定するために IdcCommandX オブジェクトにより使用されます。Site Studio コンポーネントでは、コンテンツ・サーバーが再起動されるたびにこれらの値をチェックして、最新の状態を維持します。

11.3 Cookie

Site Studio では、Cookie を使用して、contributorMode および previewId 変数の状態を伝達します。Site Studio の ASP は、HTTP を直接経由せずに IdcCommandX を使用してコンテンツ・サーバーと通信することにより動作するため、手動で Cookie の値をコンテンツ・サーバーに渡し、サーバーから戻ってくるときにそれらの値を再度取得する必要があります。これを行うには、リクエストから HTTP_COOKIE の値を取得し、addExtraHeaders 関数を使用して IdcCommandX 経由でそれらの値を渡します。レスポンス時に、SOAP レスポンスから値を取得し、必要に応じてそれらの値を HTTP レスポンス Cookie に直接追加します。

11.4 get_page.asp の操作

Site Studio ASP サイトのページを取得するリンクを作成すると、そのリンクは最終的に常に get_page.asp ファイルを示す URL になります。このファイルを ASP エンジンで実行すると、次のアクションが発生します。

まず、コンテンツ・サーバーに関する基本情報を収集するために、コンテンツ・サーバーにコールを戻します（これは、get_page.asp の最初の実行でのみ行われます）。次の情報がこのコールで収集され、ASP の Application オブジェクトに格納されます。

- コンテンツ・サーバーにより使用されている認証タイプ（内部認証、外部認証、ADSI、NTLM、LDAP など）
- フラグメントを格納しているフォルダへの相対パス
- すべての Web サイトのルートへの Web 相対パス
- コンテンツ・サーバーの相対 Web ルート
- コンテンツ・サーバーの完全な CGI パス
- コンテンツ・サーバーの完全なサーバー名

その後、get_page.asp コードは、既知のログイン・ユーザーがいるかどうかをチェックします。この時点でユーザーがいない場合、コードは処理を継続します。サイトの大部分は公開されているという仮定に基づき、保護されたリソースがリクエストされるまで、コードは既知のユーザーが存在しない状態で処理を継続します。保護されたリソースがリクエストされた時点で、ログインが表示されます（NTLM の場合は、ブラウザとサーバー間で表示なしの認証が行われます）。

get_page.asp の次のアクションは、コンテンツ・サーバーの SS_GET_PAGE サービスのコールです。コールには、リクエストが ASP サイトから送信されたことを示し、コンテンツ・サーバーに対してページを一度に編成するのに必要なすべての情報を収集するよう指示する追加のパラメータが割り当てられます。（HCSP ページでは、SS_GET_PAGE により使用するページが決定され、ページがブラウザに戻される際に、Idoc の評価により様々な情報がすべてまとめて取得されます。）

SS_GET_PAGE サービスが戻されると、最初に HTTP レスポンス Cookie に設定する必要のある SOAP レスポンス内のすべての値がチェックされます（11-5 ページの「Cookie」を参照）。これが終了すると、コードはリダイレクトが必要かどうかをチェックし、必要な場合はリダイレクトを実行して終了します。リダイレクトが必要ない場合、ページの作成に必要なすべてのデータを SOAP レスポンスから抽出し、ASP の Session に配置します。

次のデータは、ページ全体に関連します。

- ssIsLoggedIn: ログイン・ユーザーが存在するかどうか
- nodeId: 現在のノードの ID
- websiteId: 現在のサイトの ID
- isSecondaryPage: 動作するページがセカンダリ・ページであるかどうか
- userLanguageId: ユーザーのロケール識別子（en、fr、jp など）
- ssServerRelativeSiteRoot: ドメイン・サイト用の「/」と、ドメイン以外のサイト用の /<root folder>/

- ssUrlPageName: レイアウトの dDocName のかわりとなる URL の最後のページ名
- ssCacheControl: プロキシ・サーバーで使用（ページのキャッシュ期間を設定可能）

次のデータは、各リージョンに関連します。

- _dDocName: リージョンのデータ・ファイル（またはネイティブ・ドキュメント）のコンテンツ ID
- _path: weblayout 領域のデータ・ファイルへのローカル・ファイル・システム・パス
- _isNativeDoc: ドキュメントが XML データ・ファイルとネイティブ・ドキュメントのどちらであるか

これらのリージョン値は、Site Studio の ASP ディクショナリ・オブジェクトに格納され、ssGetVar 関数の使用により取得されます。リージョン間の競合を避けるため、各値の先頭には regionId が追加されます。たとえば、region2 のデータ・ファイルへのローカル・パスを取得する場合、次のコードを使用します。

```
Dim dataFileLocalPath
dataFileLocalPath = ssGetVar( "region2_path" )
```

get_page.asp コードの最後のアクションは、SS_GET_PAGE サービス・コールで識別されたレイアウト・ページでの Server.execute の実行です。これにより、すべての ASP コードが実行され、ページの様々な要素が編成されるのに応じて、必要なセッション・オブジェクトからリージョン・データ（およびその他の情報）がすべて取得されます。

11.5 ASP のマークアップの違い

HCSP レイアウト・ページでは、ページで使用されるキャラクタ・セットを識別するキャラクタ・セット宣言がページの先頭近くに存在します。これ以外に、ASP ページでは、次のような行が存在します。

```
<%Session.CodePage=65001%>
```

これは、ASP の Session オブジェクトが、ページで使用されるコードを認識するために必要です。（特に、この情報は、常に UTF-8 として格納される Site Studio の XML データ・ファイルからコンテンツを抽出する際に必要です。つまり、ASP ページでは、UTF-8 以外のページにコンテンツが抽出される場合にデータに適用する変換の種類を認識する必要があります。）

デザイナ・アプリケーションでは、この行が自動的に追加され、HTML のキャラクタ・セット宣言と同期するように管理されます（HTML の head セクションでも同様です）。デザイナの「Source」ビューでコードを編集する場合、この部分を破損しないように注意してください。

次の 3 つのタイプのマークアップがあります。

- 11-7 ページの「Site Studio リージョンのマークアップ」
- 11-8 ページの「Site Studio 要素のマークアップ」
- 11-9 ページの「フラグメントのマークアップ」

11.5.1 Site Studio リージョンのマークアップ

次に、Region1 という単一のリージョンを含む HCSP レイアウト・ページの通常のマークアップを示します。

```
<!-- SS_BEGIN_OPENREGIONMARKER(region1)-->
<!--$SS_REGIONID="region1"-->
<!--$include ss_open_region_definition -->
<!-- SS_END_OPENREGIONMARKER(region1)-->

<!-- SS_BEGIN_CLOSEREGIONMARKER(region1)-->
<!--$include ss_close_region_definition -->
<!-- SS_END_CLOSEREGIONMARKER(region1)-->
```

ASP レイアウトでの同じリージョンは、次のようになります。

```
<!-- SS_BEGIN_OPENREGIONMARKER(region1)-->
<%SS_REGIONID="region1"%>
<%=aspOpenRegionDefinition(SS_REGIONID)%>
<!-- SS_END_OPENREGIONMARKER(region1)-->

<!-- SS_BEGIN_CLOSEREGIONMARKER(region1)-->
<%=aspCloseRegionDefinition("region1")%>
<!-- SS_END_CLOSEREGIONMARKER(region1)-->
```

最初の違いは、構文が Idoc スクリプト（$）ではなく ASP（%）であることです。さらに、Idoc には、ss_open_region_definition という Idoc リソースのインクルードが存在しています。ページは Idoc の単一のコンテキスト内で編成および実行されるため、インクルードは単純にページに配置することが可能であり、ページのすべての変数はインクルードで使用できます。（この例では、SS_REGIONID の値をインクルードされたリソースで使用できます。）

しかし、ASP にはこのようなインクルード・メカニズムはありません。かわりに、次のように asp_includes.asp の事前定義された関数に対する関数コールが存在します（この関数では、リージョン ID をパラメータとして使用します）。

```
<%=aspOpenRegionDefinition(SS_REGIONID)%>
```

次に、この関数は、指定されたリージョン ID を使用して、ASP の Session オブジェクトでリージョンに必要なすべての情報を検索します。この情報は、取得するレイアウト・ページを検出するためにコンテンツ・サーバーに戻された get_page.asp の初期コールですべて収集されています。

同様のコードは、終了リージョン・マーカーにも存在し、HCSP レイアウトのインクルードが ASP レイアウトの関数コールで置き換えられています。

11.5.2 Site Studio 要素のマークアップ

次に、element1 という単一の WYSIWYG 要素に対する通常の HCSP スタイルのマークアップを示します。

```
<!--SS_BEGIN_ELEMENT(region1_element1)-->
<!--$ssIncludeXml(SS_DATAFILE,region1_element1 & "/node()")-->
<!--SS_END_ELEMENT(region1_element1)-->
```

ASP レイアウトでの同じ要素は、次のようにマークアップされます。

```
<!--SS_BEGIN_ELEMENT(region1_element1)-->
<%=aspIncludeXml(SS_REGIONID, region1_element1 & "/node()")%>
<!--SS_END_ELEMENT(region1_element1)-->
```

HCSP ページでは、特定のリージョン用のコントリビュータ・データ・ファイルの名前が、そのページに到達するのに使用されるプライマリまたはセカンダリ URL から取得されます。この名前は、ページ上で SS_DATAFILE として使用できます。

ASP サイトでは、プライマリおよびセカンダリ URL からのデータは、リージョンごとに ASP の Session オブジェクトに格納されます。そのため、リージョン ID を aspIncludeXml 関数に渡して、リージョンに関連付けられたコントリビュータ・データ・ファイルの場所を特定する必要があります。

get_page.asp ファイルからコンテンツ・サーバーに渡される初期コールでは、戻される情報に、ページで使用されるすべてのコントリビュータ・データ・ファイルへの完全なファイル・システム・パスが含まれます。つまり、初期コールが完了すると、ASP ページでは、ファイル・システム（コンテンツ・サーバーの weblayout 領域）から直接コントリビュータ・データ・ファイルにアクセス可能となり、コンテンツ・サーバーへのサービス・コールを使用してファイルを取得せずに済みます。

ページのリージョンおよび要素ごとに、対応するエントリがページの head セクションの <ssinfo> XML データ・アイランドに存在します。次に、HCSP ページのコード例の一部を示します。

```
<region id="region1" name="region1" flags="11111111" … >
<!--$region1_ACTIONS="EIPRS", region1_DCCOMMAND = … -->
<element id="region1_element1" name="element1" … >
<!--$region1_element1="region1/element1" -->
<description>element1</description>
</element>
</region>
```

ASP での同じコードは次のとおりです。

```
<region id="region1" name="region1" flags="11111111" … >
<![CDATA[<%
Call ssPutVar("region1_ACTIONS","EIPRS")
Call ssPutVar("region1_DCCOMMAND", ….)
%>]]>
<element id="region1_element1" name="element1" … >
<![CDATA[<% region1_element1="region1/element1" %>]]>
<description>element1</description>
</element>
</region>
```

ここでの大きな違いは、HCSP の場合、region1_ACTIONS や region1_element1 などの様々なリージョン・パラメータがページ上で Idoc 変数として作成されるため、それらの変数を後からページの Idoc スクリプトで使用できることです。ASP では、同じ変数は Site Studio の ASP ヘルパー関数 ssPutVar を通じて ASP の Session オブジェクトに記述されるため、それらの変数は ASP エンジンによるページの編成時に ASP エンジンで使用できます。

ASP バージョンでは、すべての ASP コードを CDATA セクションに配置して、通常であれば XML ファイルで無効となるすべての文字のエスケープ処理を回避しています。

11.5.3 フラグメントのマークアップ

フラグメントは、レイアウト・ページの複数の場所でマークアップされます。レイアウト・ページの head セクションの <ssinfo> XML データ・アイランドには、各フラグメントのエントリがあります。これにより、フラグメントが識別され、フラグメントの取得元のライブラリとフラグメントのパラメータが判別されます。次に、フラグメントを構成する各スニペットのエントリがあります。

HCSP ページの標準フラグメントの Explorer Menu Bar（『Oracle Universal Content Management Site Studio デザイナ・ガイド』を参照）では、<ssinfo> XML データ・アイランドに次のエントリが含まれます。

```
<fragmentinstance id="fragment1" fragmentid="ExplorerMenuBar" library =
"server:SS_Fragments_Dynamic">
<!--$fragment1_ssShowHome="true" -->
</fragmentinstance>
```

ASP レイアウトの同等のフラグメント（ASP Explorer Menu Bar）には、次のエントリが含まれます。

```
<fragmentinstance id="fragment1" fragmentid="asp_ExplorerMenuBar" library
="server:SS_Fragments_ASP">
<![CDATA[<% Call ssPutVar("fragment1_ssShowHome","true") %>]]>
</fragmentinstance>
```

前述の項におけるリージョン変数の処理と同じ方法で、フラグメント変数は ssPutVar 関数を通じて ASP の Session オブジェクトに配置されます（HCSP のようにページ・リンクでは宣言されません）。これらのフラグメントには、2 つのスニペットがあります。

最初に HCSP バージョンを示します。

```
<!-- SS_BEGIN_SNIPPET(fragment1,1)-->
<!--$ssFragmentInstanceId="fragment1", ssIncludeXml("SS_Fragments_Dynamic",
"fragments/fragment[@id='ExplorerMenuBar']/snippets/snippet[@id='1']/text()")-->
<!-- SS_END_SNIPPET(fragment1,1)-->
<!-- SS_BEGIN_SNIPPET(fragment1,2)-->
<!--$ssFragmentInstanceId="fragment1", ssIncludeXml("SS_Fragments_Dynamic",
"fragments/fragment[@id='ExplorerMenuBar']/snippets/snippet[@id='2']/text()")-->
<!-- SS_END_SNIPPET(fragment1,2)-->
```

次に ASP バージョンを示します。

```
<!-- SS_BEGIN_SNIPPET(fragment1,1)-->
<%Call ssPutVar("ssFragmentInstanceId","fragment1")%>
<link rel="stylesheet" type="text/css" href="<%=
HttpRelativeFragmentsRoot%>asp_explorermenubar/asp_ExplorerMenuBar.css" />
<script src="<%=HttpRelativeFragmentsRoot%> asp_explorermenubar/asp_ExplorerMenuBar.js"
type="text/javascript">
</script>
<!-- SS_END_SNIPPET(fragment1,1)-->
<!-- SS_BEGIN_SNIPPET(fragment1,2)-->
<%Call ssPutVar("ssFragmentInstanceId","fragment1")%>
<%Server.execute(HttpRelativeFragmentsRoot &
"asp_explorermenubar/asp_ExplorerMenuBar.asp")%>
<!-- SS_END_SNIPPET(fragment1,2)-->
```

これらのコードはまったく異なります。これは、ASP ではすべてのスニペットがインライン・スニペットとしてインクルードされるのに対し、一般的に HCSP スニペットは参照によりインクルードされることが原因です。

HCSP バージョンの ssIncludeXml コールでは、参照によるインクルード・メカニズムが示されているのに対し、ASP ではページに直接インライン配置されたスニペットの実際のコードが示されています。ASP スニペットをこのようにインクルードする理由はいくつかありますが、詳細は次の項で説明します。

11.6 ASP フラグメント

ASP フラグメントの詳細を検討する前に、典型的な HCSP フラグメントがどのように動作し、ページでインクルードされるかを確認しておきます。通常、HCSP フラグメント・スニペットは、参照によりページでインクルードされます。そのため、ページでは、次のようなコードを確認できます。

```
<!-- SS_BEGIN_SNIPPET(fragment1,1)-->
<!--$ssFragmentInstanceId="fragment1", ssIncludeXml("SS_Fragments_Dynamic",
"fragments/fragment[@id='ExplorerMenuBar']/snippets/snippet[@id='1']/text()")-->
<!-- SS_END_SNIPPET(fragment1,1)-->
```

コメント・タグの使用により、スニペットの開始と終了がマークされています（デザイナ・アプリケーションにより使用されるマーカー）。また、単一行の Idoc スクリプトは、ssFragmentInstanceId という変数の宣言（およびその値の fragment1 への設定）と、Idoc 関数 ssIncludeXml のコールという 2 つの処理を実行しています。

ssIncludeXml では、2 つのパラメータを使用します。使用するフラグメント・ライブラリの名前（SS_Fragments_Dynamic）と、インクルードするスニペットを識別する XPath 式（fragments/fragment[@id='ExplorerMenuBar']/snippets/snippet[@id='1']/text()）です。

ssIncludeXml は、所定の XPath により識別されたフラグメント・ライブラリの一部を抽出します。この例の場合、次のようになります。

```
<script>
{
<!--$ssShowHome = eval("<$" & ssFragmentInstanceId & "_" & "ssShowHome" & "$>")-->
    var <!--$ssFragmentInstanceId-->_ExplorerMenuBar =
    new ExplorerMenuBar(
        '<!--$ssFragmentInstanceId-->_ExplorerMenuBar',
        '<!--$HttpCgiPath-->',
        '<!--$HttpRelativeFragmentsRoot-->explorermenubar/',
        '<!--$ssShowHome-->');

     <!--$ssFragmentInstanceId-->_ExplorerMenuBar.Display(g_navNode_Root);
}
</script>
```

これは、Idoc 評価エンジンに渡され、その時点で fragmentInstanceId 変数がフラグメントのこのインスタンスに対応する実際の値で置き換えられます。次に、その結果が直接ページ上に配置されます。

ただし、ASP の場合、aspIncludeXml がフラグメント・ライブラリからコードを抽出した後に、スニペット・コードをさらに評価する方法はありません。一方で、ASP 変数を各フラグメント・スニペットに提供するため、（コントリビュータ・データ・ファイルから要素データをインクルードするのに使用できる aspIncludeXml 以外に）ページでスニペット・コードをインクルードする方法が必要です。

注意： ASP には、#INCLUDE を使用したインクルード・メカニズムが確かに存在しますが、これらのインクルードは、他のすべてのスクリプトがページで評価される前に行われます。つまり、このタイプのインクルードでは、変数を使用してインクルード対象のファイルの名前を構成することはできません。

ただし、ASP には、サーバー・オブジェクトの実行メソッドを使用して複数の ASP ファイルからページを編成する別の方法が存在します。実行メソッドを、ファイル名に対応する変数名と組み合せて使用することで、実行時に構成されるファイル名を使用できます。

これにより、前述のインクルード・メカニズムと同様のメカニズムが提供されますが、やはり変数名が必要です。主な問題点は、実行がその独自のコンテキスト内で発生するため、実行時に有効範囲内のいずれのローカル変数にもアクセスできないことです。ただし、リクエスト、レスポンス、サーバー、セッション、アプリケーションなどの主な ASP 変数は参照できます。

そのため、必要な任意のパラメータは、セッション・オブジェクトを通じて渡すことができます。

これらの問題のため、ASP フラグメント・スニペットでは、フラグメント・アセットとしてフラグメントに追加される ASP ファイルにその実装を配置することが推奨されます。これにより、実際のスニペット・コードは、アセット・ファイルを参照する server.execute 文とほとんど同じになります。

たとえば、ASP Explorer Menu Bar（デザイナ・ガイドのサンプル・フラグメントに関する項を参照）の 2 つ目のスニペットに対応する実際のスニペット・コードは、次のようになります。

```
<%Server.execute(HttpRelativeFragmentsRoot & "asp_explorermenubar/
asp_ExplorerMenuBar.asp")%>
```

スニペットのすべての実装は、実際には asp_ExplorerMenuBar.asp ファイルに格納されています。

11.7 ASP リンク関数

現在、ASP レイアウトで使用できるサーバー側リンク関数は 2 つあります（ssLink と ssNodeLink）。これらの関数は、次のように定義されています。

```
ssLink( contentId, nodeId, siteId )
ssNodeLink( nodeId, siteId )
```

どちらの関数でも、第 1 パラメータのみが必須です。ただし、ASP レイアウトで直接使用する場合は、すべてのパラメータを指定する必要があります。たとえば、レイアウト・ページで使用可能なこれらの関数のすべての書式は、次のとおりです。

```
<%=ssLink("doc1","","")%>
<%=ssLink("doc1","node2","") %>
<%=ssLink("doc1","node2","site3") %>
<%=ssNodeLink("node1","") %>
<%=ssNodeLink("node1","site2")%>
```

データ・ファイル内では、第 1 パラメータが必須であり、残りのパラメータはオプションです。

データ・ファイルで使用される場合の構文は、ASP ではありません。データ・ファイルは、任意のタイプのサイトで使用される可能性があり、言語に依存しないマークアップが使用されるためです。このマークアップは、Idoc パーサーでも認識されるため、データ・ファイルは ASP、Idoc および JSP ファイルともに同じ構文で使用できます。この構文の例は次のとおりです。

```
[!--$ssLink("doc1")--]
[!--$ssLink("doc1","node2")--]
[!--$ssLink("doc1","node2","site3")--]
[!--$ssNodeLink("node1")--]
[!--$ssNodeLink("node1","site2")--]
```

11.8 ASP データ・ファイル

ASP のデータ・ファイルは、現在、前述のサーバー側リンク関数を扱う目的で処理されるため、カスタマイズに役立つ可能性のある他のいくつかのトークンもサポートされます。サポートされるトークンは次のとおりです。

- ssServerRelativeSiteRoot
- HttpRelativeWebRoot
- HttpCgiPath
- HttpBrowserFullCgiPath
- ssLink
- ssNodeLink
- HttpRelativeFragmentsRoot
- HttpRelativeWebsitesRoot

これらの特定のトークンの詳細は、ASP のリファレンス・マニュアルを参照してください。

11.9 XML でのサイト・ナビゲーション

現在、asp_includes.asp には、サイト階層を XML DOM にロードして、カスタム・コードによる階層の問合せやカスタム・ナビゲーション・フラグメントの作成などを可能にするヘルパー関数が存在します。

ナビゲーションは、次のようにロードして使用できます。

```
Dim ssSiteNavXml
Set ssSiteNavXml = ssLoadSiteNavXml()
```

これで、xPath 式を使用して次のように階層にアクセスできます。

```
Dim node
Set node = ssSiteNavXml.selectSingleNode("//*[@id=whatNodeIdYouWant]")
```

階層の各ノードには、カスタム・ナビゲーションに役立つ次の値が含まれます（追加の属性も存在します。詳細は、コンテンツ・サーバーの weblayout/websites フォルダにある website フォルダの sitenavigation.xml ファイルを参照してください）。

- **sectionId:** Web サイト・セクションの ID
- **label:** デザイナの UI に表示されるノードのラベル
- **level:** 階層内のレベル
- **href:** サイトのルート相対パス（ssUrlPrefix と結合して完全なサーバー相対 URL を作成することが必要）

これらの属性は、次のように簡単に取得できます。

```
node.getAttribute("label")
```

サンプル・フラグメントの ASP Bread Crumb Plain には、ナビゲーション XML ファイルを使用してナビゲーション構造を構成するための方法が具体的に示されています。

11.10 getSearchResultsEx とフレンドリ URL（ssUrl）

検索結果の結果セットに追加の列（ssUrl）を戻す、検索に使用できる新規サービスがあります。ssUrl には、ターゲット・ドキュメントに対するフルパス・ベースの（フレンドリ）URL が含まれます。新規サービスには、ヘルパー関数の getSearchResultsEx を使用してアクセスします。この関数は、次のように定義されています。

```
getSearchResultsEx(ssQueryText,ssUserSearchText,ssLimitScope,ssDontShowInLists,ssTarget
NodeId,ssResultCount,ssSortOrder,ssSortField,SortSpec,StartRow,PageNumber)
```

コールの結果は、次のように searchResults 結果セットを問い合せることができる XML DOM です。

```
Dim searchResults, xPath, xpSearchResults, resultList
Set searchResults = getSearchResultsEx(…,…)
xPath = "//SOAP-ENV:Envelope/SOAP-ENV:Body/idc:service/idc:document"
xpSearchResults = xPath & "/idc:resultset[@name='SearchResults']/*"
Set resultsList = searchResults.selectNodes(xpSearchResults)
```

結果リストを取得したら、リストをループして ssUrl 値を選択することで、次のように検索のターゲットへのリンクを作成できます（この例では、リンクのラベルとして dDocTitle を使用しています）。

```
<%
For each doc In resultsList
%>
    <a href="<%=xGet(doc,"ssUrl")%>"><%=xGet(doc,"dDocTitle")%></a>
<%
Next
%>
```

この処理の正確な実行方法は、ASP Search Results Plain フラグメントを参照してください。

結果で getSearchResultsEx とパス・ベースの URL を使用する場合、パス・ベースの URL の計算は、ターゲット・ノード ID を明示的に指定するとずっと高速化します。結果にパス・ベースの URL が不要で、パフォーマンスを向上させる必要がある場合、computeFriendlyUrls の値を false または 0（ゼロ）に設定することでパス・ベースの URL の計算を無効にできます。これを行うには、getSearchResultsEx 関数をコピーして新しく customGetSearchResults などの名前を付け、fieldData 変数が構成される部分に次のコードを追加する必要があります。

```
"<idc:field name=""computeFriendlyUrls"">0</idc:field>"
```

11.11 ASP のサポート関数とサンプル・コード

asp_includes.asp ファイルには、カスタム ASP コードを作成する必要のある Web サイト設計者にとって役立つ関数が多く含まれます。頻繁に使用されるものの 1 つとして、リージョンに関連付けられたデータ・ファイルまたはネイティブ・ドキュメントの特定のメタデータを取得するためにコンテンツ・サーバーに戻されるサービス・コールがあります。ここでは、次のようにいくつかの便利な機能とともに、これらの関数について説明します。

- ページで使用可能なデータにリージョンごとにアクセスする方法
- asp_includes.asp で提供される関数を使用する方法
- コンテンツ・サーバーに戻すサービス・コールを作成する方法
- 戻された SOAP レスポンスを解釈する方法

たとえば、ページのリージョン 2 に関連付けられたデータ・ファイルから xPrice メタデータ値を取得するとします。

最初に、次のように dDocName を取得します。

```
Dim dDocName
dDocName = ssGetVar("region2_dDocName")
```

次に、SOAP リクエストを構成してコンテンツ・サーバーに渡す必要があります。これは、asp_includes.asp の次の各関数を使用して行います。

```
Dim fieldData, requestXml, soapResponse, xpDocInfo, docInfoRSet, docNode

fieldData = "<idc:field name=""dDocName"">" & dDocName & "</idc:field>"
requestXml = createRequestXml(fieldData, false)
Set soapResponse = executeService("DOC_INFO_BY_NAME", requestXml)
If Not soapResponse.documentElement Is Nothing Then
   Set docNode = soapResponse.selectSingleNode(xPathDocNode)
   If Not docNode Is Nothing Then
      xpDocInfo = xPathDocNode & "/idc:resultset[@name='DOC_INFO']/*"
      Set docInfoRSet = soapResponse.selectNodes(xpDocInfo)
      For Each record In docInfoRSet
        xPriceValue = xGet(record,"xPrice")
      Next
   End If
End If
```

このリクエスト用の特定のフィールドを fieldData 文字列に追加し、createRequestXml 関数を使用してこれらの特定のフィールドを SOAP リクエストに必要とされる標準フィールドに結合しています。

executeService コールからのレスポンスは、XML DOM オブジェクトです。最初に、レスポンス内に少なくともドキュメント要素が含まれていることをチェックします。含まれていない場合、実行サービスで何か問題が発生しています。documentElement が存在する場合、次のように定義された xPathDocNode により識別されるノードの取得を試みます。

```
//SOAP-ENV:Envelope/SOAP-ENV:Body/idc:service/idc:document
```

このノードから、すべてのフィールド・データを取得できます。フィールド・データは、通常の（SOAP 以外の）コンテンツ・サーバー・リクエストに対する hda レスポンスに含まれるローカル・プロパティと同じです。

レスポンスには、XML 階層のドキュメント・オブジェクトの下にある様々な結果セットも含まれます。ここでは、次のようにアクセスできる DOC_INFO 結果セットを検索しています。

```
/idc:resultset[@name='DOC_INFO']/*
```

これは、ドキュメント要素の下の xPath であるため、次のように 2 つの部分を結合することで xPath 全体を構成できます。

```
xpDocInfo = xPathDocNode & "/idc:resultset[@name='DOC_INFO']/*"
```

結果セットを取得したら、ループを使用してその中のレコードを反復処理します。DOC_INFO_BY_NAME では、結果セット内の 1 つのエントリのみを対象としますが、このループは別の方法を示しています。

最後に、xGet ヘルパー関数を使用して、DOC_INFO から xPrice メタデータ・フィールドの値を取得しています。

12

マネージャ設定ファイル

マネージャ設定ファイルは、サイト設計者がマネージャで使用可能な機能を制御する際に使用できる複数の構成オプションが含まれた XML ファイルです。

設計者は、「Site Assets」ペインでファイルを編集するときに表示される「Form」ビューを使用して、このファイルの複数の設定を変更できます。「Source」ビューでこのファイルを編集すると、より高度な設定を操作できます。

この章の内容は次のとおりです。

12.1「Source」ビューのマネージャ設定ファイル

マネージャ設定ファイルは、「Source」ビューで次のように表示されます。

```
<ssm:settings>
     <ssm:general />
     <ssm:addSection>
          <ssm:urlDirName />
          <ssm:id />
          <ssm:primaryUrl />
          <ssm:secondaryUrl />
          <ssm:secondaryUrlVariableField />
     </ssm:addSection>
     <ssm:removeSection />
     <ssm:moveSection />
     <ssm:setErrorHandler />
     <ssm:editProperties>
          <ssm:id />
          <ssm:label />
          <ssm:active />
          <ssm:contributorOnly />
          <ssm:urlDirName />
          <ssm:urlPageName />
          <ssm:maxAge />
     </ssm:editProperties>
     <ssm:editCustomProperties />
     <ssm:primaryLayout >
          <ssm:presentation></ssm:presentation>
          <ssm:queryText></ssm:queryText>
     </ssm:primaryLayout>
     <ssm:secondaryLayout>
          <ssm:presentation></ssm:presentation>
          <ssm:queryText></ssm:queryText>
     </ssm:secondaryLayout>
     <ssm:sectionOverride>
     </ssm:sectionOverride>
</ssm:settings>
```

12.2 <ssm:settings> タグ

<ssm:settings> タグは、マネージャ設定ファイル内のルート XML 要素です。

属性

- **xmlns:** この XML ファイルの名前空間。この属性の値は常に次のようになります。

  ```
  xmlns:ssm="http://www.stellent.com/sitestudio/managersettings/"
  ```

子タグ

<ssm:settings> タグは、次のオプションの子タグを格納できます。

- 12-3 ページの「<ssm:general> タグ」
- 12-4 ページの「<ssm:addSection> タグ」
- 12-4 ページの「<ssm:removeSection> タグ」
- 12-5 ページの「<ssm:moveSection> タグ」
- 12-5 ページの「<ssm:setErrorHandler> タグ」
- 12-5 ページの「<ssm:editProperties> タグ」
- 12-6 ページの「<ssm:editCustomProperties> タグ」
- 12-6 ページの「<ssm:primaryLayout> タグ」
- 12-7 ページの「<ssm:secondaryLayout> タグ」
- 12-7 ページの「<ssm:sectionOverride> タグ」

12.3 <ssm:general> タグ

<ssm:general> タグには、汎用の構成設定が含まれます。

属性

- **contributorOnly:** true に設定すると、マネージャ・アプリケーションはブラウザがコントリビューション・モードの場合にのみ表示されます。false に設定すると、マネージャ・アプリケーションは、コントリビューション・モードとコンシューマ・モードの両方で表示されます。
- **autoManage:** true に設定すると、マネージャ・アプリケーションは、最初に表示されたときに自動的にサーバーに接続し、サイトの管理に必要な情報を収集します。false に設定すると、マネージャ・アプリケーションは、サーバーと通信する前にユーザーが「(manage site)」リンクをクリックするのを待機します。
- **hierarchy:** マネージャ・アプリケーションでの階層の表示を制御します。次のいずれかの値を指定する必要があります。
  - **hide:** サイト階層を表示しません。
  - **showAll:** サイト階層全体を表示します。
  - **showCurrentSectionOnly:** 現在のセクション（およびその子）のみを表示します。
  - **displayConsole:** true に設定すると、マネージャ・アプリケーションと Site Studio コンポーネント間のすべてのサーバー通信をリストしたコンソールが表示されます。デバッグ目的専用です。
  - **updateTitle:** true に設定すると、現在選択されているセクションおよびアクション・タブを使用してブラウザのタイトル・バーが更新されます。false に設定すると、タイトル・バーは更新されません。
  - **updateUrl:** true に設定すると、現在選択されているセクションおよびアクション・タブの状態を使用して URL が更新されます。false に設定すると、URL は更新されません。

子タグ

<ssm:general> タグには、子タグは含まれません。

12.4 <ssm:addSection> タグ

<ssm:addSection> タグには、新規セクション追加機能の構成設定が含まれます。

属性

- **hidden:** true に設定すると、マネージャからこの機能が隠蔽されます。

子タグ

<ssm:addSection> タグは、次のオプションの子タグを格納して、新規セクション追加機能の動作を制御できます。

- **<ssm:urlDirName>** には、次のオプション属性が含まれます。
  - **hidden:** true または false
  - **disabled:** true または false
- **<ssm:id>** には、次のオプション属性が含まれます。
  - **hidden:** true または false
  - **disabled:** true または false
- **<ssm:primaryUrl>** には、次のオプション属性が含まれます。
  - **inherit:** true または false に設定し、新規セクションでその親セクションのプライマリ URL を継承するかどうかを指定します。
  - **default:** 新規セクションに primaryUrl のデフォルト値を提供します。
- **<ssm:secondaryUrl>** には、次のオプション属性が含まれます。
  - **inherit:** true または false に設定し、新規セクションでその親セクションのセカンダリ URL を継承するかどうかを指定します。
  - **default:** 新規セクションに secondaryUrl のデフォルト値を提供します。
- **<ssm:secondaryUrlVariableField>** には、次のオプション属性が含まれます。
  - **inherit:** true または false に設定し、新規セクションでその親セクションの secondaryUrlVariableField を継承するかどうかを指定します。
  - **default:** 新規セクションに secondaryUrlVariableField のデフォルト値（true または false）を提供します。

12.5 <ssm:removeSection> タグ

<ssm:removeSection> タグには、セクション削除機能の構成設定が含まれます。

属性

- **hidden:** true に設定すると、マネージャからこの機能が隠蔽されます。

子タグ

<ssm:removeSection> タグには、子タグは含まれません。

12.6 <ssm:moveSection> タグ

<ssm:moveSection> タグには、セクション移動機能の構成設定が含まれます（ドラッグ・アンド・ドロップによる移動を含む）。

属性

- **hidden:** true に設定すると、マネージャからこの機能が隠蔽されます。

子タグ

<ssm:moveSection> タグには、子タグは含まれません。

12.7 <ssm:setErrorHandler> タグ

<ssm:setErrorHandler> タグには、エラー・ハンドラ設定機能の構成設定が含まれます。

属性

- **hidden:** true に設定すると、マネージャからこの機能が隠蔽されます。

子タグ

<ssm:setErrorHandler> タグには、子タグは含まれません。

12.8 <ssm:editProperties> タグ

<ssm:editProperties> タグには、標準のセクション・プロパティを編集するための構成設定が含まれます。

属性

- **hidden:** true に設定すると、マネージャからこの機能が隠蔽されます。

子タグ

<ssm:editProperties> タグは、次のオプションの子タグを格納できます。

- **<ssm:id>** には、次のオプション属性が含まれます。
  - **hidden:** true または false
  - **disabled:** true または false
- **<ssm:label>** には、次のオプション属性が含まれます。
  - **hidden:** true または false
  - **disabled:** true または false
- **<ssm:active>** には、次のオプション属性が含まれます。
  - **hidden:** true または false
  - **disabled:** true または false
- **<ssm:contributorOnly>** には、次のオプション属性が含まれます。
  - **hidden:** true または false
  - **disabled:** true または false
- **<ssm:urlDirName>** には、次のオプション属性が含まれます。
  - **hidden:** true または false
  - **disabled:** true または false

- **<ssm:urlPageName>** には、次のオプション属性が含まれます。
  - **hidden:** true または false
  - **disabled:** true または false
- **<ssm:maxAge>** には、次のオプション属性が含まれます。
  - **hidden:** true または false
  - **disabled:** true または false

12.9 <ssm:editCustomProperties> タグ

<ssm:editCustomProperties> タグには、カスタム・プロパティ機能の構成設定が含まれます。

属性

- **hidden:** true に設定すると、マネージャからこの機能が隠蔽されます。

子タグ

<ssm:editCustomProperties> タグには、子タグは含まれません。

12.10 <ssm:primaryLayout> タグ

<ssm:primaryLayout> タグには、プライマリ・レイアウト選択機能の動作を制御するための構成設定が含まれます。

属性

- **hidden:** true に設定すると、マネージャ・アプリケーションで「Layout」タブが非表示になります。
- **externalHidden:** true に設定すると、セクションのプライマリ・レイアウトとして外部 URL を選択する機能が非表示になります。
- **previewHidden:** true に設定すると、「Layout」タブのプレビュー IFrame が非表示になります。

子タグ

<ssm:primaryLayout> タグは、次のオプションの子タグを格納できます。

- **<ssm:presentation>** には、単純な HTML プレゼンテーション文字列を提供する CDATA セクションが含まれます。これにより、レイアウト選択コンボ・ボックスにレイアウト情報を表示できます。単純なインライン HTML は、$field$ としてマークされたメタデータ・フィールド宣言を通じて可能となります。次にいくつかの例を示します。
  - $dDocTitle$
  - $dDocName$
  - $dDocTitle$（$dDocName$）
  - $dDocName$
- **<ssm:queryText>** には、レイアウト選択コンボ・ボックスにデータを移入するための問合せ文字列を提供する CDATA セクションが含まれます。また、このタグは、次のオプション属性も格納できます。
  - **limitscope:** true に設定すると、検索を現在の Web サイトのアイテムのみに制限して実行できます。

12.11 <ssm:secondaryLayout> タグ

<ssm:secondaryLayout> タグには、セカンダリ・レイアウト選択機能の動作を制御するための構成設定が含まれます。

属性

- **hidden:** true に設定すると、マネージャ・アプリケーションで「Secondary Layout」タブが非表示になります。
- **previewHidden:** true に設定すると、「Secondary Layout」タブのプレビュー IFrame が非表示になります。

子タグ

<ssm:secondaryLayout> タグは、次のオプションの子タグを格納できます。

- **<ssm:presentation>** には、単純な HTML プレゼンテーション文字列を提供する CDATA セクションが含まれます。これにより、レイアウト選択コンボ・ボックスにレイアウト情報を表示できます。単純なインライン HTML は、$field$ としてマークされたメタデータ・フィールド宣言を通じて可能となります。次にいくつかの例を示します。
  - $dDocTitle$
  - $dDocName$
  - $dDocTitle$（$dDocName$）
  - $dDocName$
- **<ssm:queryText>** には、レイアウト選択コンボ・ボックスにデータを移入するための問合せ文字列を提供する CDATA セクションが含まれます。また、このタグは、次のオプション属性も格納できます。
  - **limitscope:** true に設定すると、検索を現在の Web サイトのアイテムのみに制限して実行できます。

12.12 <ssm:sectionOverride> タグ

<ssm:sectionOverride> タグには、前述のデフォルト設定を上書きする特定の Web サイト・セクションの構成設定が含まれます。

属性

- **nodeId:** 上書きされるセクションの ID

子タグ

<ssm:sectionOverride> タグは、次のオプションの子タグを格納できます。これらの子タグの詳細は、次の各項で説明済です。

- 12-4 ページの「<ssm:addSection> タグ」
- 12-4 ページの「<ssm:removeSection> タグ」
- 12-5 ページの「<ssm:moveSection> タグ」
- 12-5 ページの「<ssm:setErrorHandler> タグ」
- 12-5 ページの「<ssm:editProperties> タグ」
- 12-6 ページの「<ssm:editCustomProperties> タグ」
- 12-6 ページの「<ssm:primaryLayout> タグ」
- 12-7 ページの「<ssm:secondaryLayout> タグ」

<ssm:sectionOverride> 内のサブセクションの属性およびコンテンツは、一般的な方法で使用できますが、次の例外があります。

- <ssm:moveSection> は、上書きされるセクションにのみ適用される次の 2 つの追加属性を格納できます。
  - **source:** false に設定すると、このセクションは移動アクションのソースとして使用されません。
  - **target:** false に設定すると、このセクションは移動アクションのターゲットとして使用されません。

12.13 マネージャ設定ファイルの例

```
<?xml version='1.0' encoding='utf-8' ?>
<ssm:settings xmlns:ssm="http://www.stellent.com/sitestudio/managersettings/">
     <ssm:general contributorOnly="false"
          autoManage="true"
          hierarchy="showAll"
          displayConsole="false"
          updateTitle="true"
          updateUrl="true" />

     <ssm:addSection hidden="false">
          <ssm:urlDirName hidden="false" disabled="false" />
          <ssm:id hidden="false" disabled="false" />
          <ssm:primaryUrl inherit="true" default="" />
          <ssm:secondaryUrl inherit="true" default="" />
          <ssm:secondaryUrlVariableField inherit="true" default="" />
     </ssm:addSection>

     <ssm:removeSection hidden="false" />
     <ssm:moveSection hidden="false" />
     <ssm:setErrorHandler hidden="false" />

     <ssm:editProperties hidden="false" >
          <ssm:id hidden="false" disabled="true" />
          <ssm:label hidden="false" disabled="false" />
          <ssm:active hidden="false" disabled="false" />
          <ssm:contributorOnly hidden="false" disabled="false" />
          <ssm:urlDirName hidden="false" disabled="false" />
          <ssm:urlPageName hidden="false" disabled="false" />
          <ssm:maxAge hidden="false" disabled="false" />
     </ssm:editProperties>

     <ssm:editCustomProperties hidden="false" />

     <ssm:primaryLayout hidden="false" externalHidden="false" previewHidden="false">
          <ssm:presentation>
               <![CDATA[$dDocTitle$ (<i>$dDocName$</i>)]]>
          </ssm:presentation>
          <ssm:queryText limitScope="true">
               <![CDATA[]]>
          </ssm:queryText>
     </ssm:primaryLayout>

     <ssm:secondaryLayout hidden="false" previewHidden="false">
          <ssm:presentation>
               <![CDATA[$dDocTitle$ (<i>$dDocName$</i>)]]>
          </ssm:presentation>
          <ssm:queryText limitScope="true">
               <![CDATA[]]>
          </ssm:queryText>
```

```
        </ssm:secondaryLayout>

        <ssm:sectionOverride nodeId="42">
             <ssm:addSection hidden="true" />
             <ssm:removeSection hidden="true" />
             <ssm:moveSection hidden="true" />
             <ssm:setErrorHandler hidden="true" />
             <ssm:setErrorHandler hidden="true" />
             <ssm:editProperties hidden="false" />
             <ssm:editCustomProperties hidden="true" />
             <ssm:primaryLayout hidden="true" />
             <ssm:secondaryLayout hidden="true" />
        </ssm:sectionOverride>

    </ssm:settings>
```

13

Content Tracker 統合

Site Studio と Content Tracker を組み合せて使用し、Site Studio Web サイトに関するレポートを生成できます。2 つの製品の統合は、複数の構成フラグを使用してカスタマイズできます。構成フラグでは、レポート対象のデータを制御します。

Content Tracker（データ・エンジン・コントロール・センター）を使用して、追跡対象のデータと、各サービス・コールでそのデータを追跡する場所を制御できます。詳細は、『Oracle Universal Content Management Content Tracker 管理ガイド』を参照してください。

この章の内容は次のとおりです。

13.1 追跡対象データ

追跡されるサイト・アクセスには次の 2 つのタイプがあります。

- **階層アクセス :** 一般的にセクション ID により行われるアクセスで、セクションの実際のコンテンツにかかわらずそのセクションを対象とするアクセスです。
- **コンテンツ・アクセス :** コンテンツが実際に表示される場所にかかわらず、特にコンテンツの一部を対象とするアクセスです。

階層アクセス	コンテンツ・アクセス	SS 変数	CT フィールド名
レイアウト・ページの dDocName	データ・ファイル、ネイティブ・ドキュメントまたはフラグメント・ライブラリの dDocName	targetPage	sc_scs_dDocName
レイアウト・ページの dID	データ・ファイル、ネイティブ・ドキュメントまたはフラグメント・ライブラリの dID	targetdID	sc_scs_dID
サイト ID	サイト ID	targetSiteId	extField_1
ノード ID	ノード ID	targetNodeId	extField_2
セカンダリ・ページ	セカンダリ・ページ	targetIsSecondary	extField_3
Web サイトのオブジェクト・タイプ	Web サイトのオブジェクト・タイプ	targetWebsiteObjectType	extField_4
コントリビューション・モード	コントリビューション・モード	SSContributor	extField_5
（未使用）	データ・ファイル、ネイティブ・ドキュメントまたはフラグメントが使用されているレイアウトの dDocName	targetContentId	extField_6
サイト相対 URL	サイト相対 URL	siteRelativeUrl	extField_7

エラーが発生すると、次の値が Content Tracker データベースに格納されます。

エラー・タイプ	SS 変数	CT フィールド名
サーバーで取得して解決できなかった元のサイト相対 URL	originalSiteRelativeUrl	extField_8
元の URL が無効であったかどうか	invalidSiteRelativeUrl	extField_9

サイトにエラー・ハンドラ・ページがあり、そのページにアクセスできる場合、siteRelativeUrl はそのエラー・ページの URL に設定されます。エラーの原因となった元の URL は、originalSiteRelativeUrl 値として設定されます。invalidSiteRelativeUrl 値は、1 に設定されます。

サイトにエラー・ハンドラ・ページがない状態でエラーが発生すると、元の URL が siteRelativeUrl として記録され、invalidSiteRelativeUrl 値は 1 に設定されます。この場合、originalSiteRelativeUrl 値は空になります。

13.2 構成フラグ

次の構成フラグを config.cfg で設定することにより、Site Studio で追跡するデータと、その情報の追跡方法を制御します。

- `SSTrackContentAccess=[true|false]`

  これにより、コンテンツ・アクセスを有効化するかどうかを指定します。デフォルト設定は true です。false に設定すると、階層アクセスのみが追跡されます。

- `SSTrackFragmentAccess=[true|false]`

  これにより、フラグメント・ライブラリに対するアクセスを記録するかどうかを指定します。デフォルト設定は false です。

- `SSTrackerReportNumDaysBack=[n|7]`

  これにより、コントリビューション・アイコンのポップアップ・メニューを使用してアクセスできるコンテンツ・アクセス・タイプのレポートの履歴日数を指定します。デフォルト設定は 7 であり、n は正数（日数）です。

A

サード・パーティ・ライセンス

この付録では、この製品に含まれるすべてのサード・パーティ製品のサード・パーティ・ライセンスを示します。

A.1 Apache Software License

```
* Copyright 1999-2004 The Apache Software Foundation.
* Licensed under the Apache License, Version 2.0 (the
* "License"); you may not use this file except in compliance
* with the License.
* You may obtain a copy of the License at
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing,
* software distributed under the License is distributed on an
* "AS IS" BASIS,WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
* either express or implied.
* See the License for the specific language governing
* permissions and limitations under the License.
```

A.2 W3C Software Notice and License

```
* Copyright 1994-2000 World Wide Web Consortium,
* (Massachusetts Institute of Technology, Institut National de
* Recherche en Informatique et en Automatique, Keio University).
* All Rights Reserved.  http://www.w3.org/Consortium/Legal/
*
* This W3C work (including software, documents, or other related
* items) is being provided by the copyright holders under the
* following license. By obtaining, using and/or copying this
* work, you (the licensee) agree that you have read, understood,
* and will comply with the following terms and conditions:
*
* Permission to use, copy, modify, and distribute this software
* and its documentation, with or without modification, for any
* purpose and without fee or royalty is hereby granted, provided
* that you include the following on ALL copies of the software
* and documentation or portions thereof, including
* modifications, that you make:
*
*   1. The full text of this NOTICE in a location viewable to
* users of the redistributed or derivative work.
*
*   2. Any pre-existing intellectual property disclaimers,
* notices, or terms and conditions. If none exist, a short
* notice of the following form (hypertext is preferred, text is
* permitted) should be used within the body of any redistributed
* or derivative code: "Copyright  [$date-of-software] World
* Wide Web Consortium, (Massachusetts Institute of Technology,
* Institut National de Recherche en Informatique et en
* Automatique, Keio University). All Rights Reserved.
*  http://www.w3.org/Consortium/Legal/"
*
*   3. Notice of any changes or modifications to the W3C files,
* including the date changes were made. (We recommend you
* provide URIs to the location from which the code is derived.)
*
* THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND
* COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES,
* EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES
* OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR
* THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT
* INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR
* OTHER RIGHTS.
*
* COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT,
```

```
* SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE
* SOFTWARE OR DOCUMENTATION.
*
* The name and trademarks of copyright holders may NOT be used
* in advertising or publicity pertaining to the software without
* specific, written prior permission. Title to copyright in this
* software and any associated documentation will at all times
* remain with copyright holders.
*
```

A.3 Zlib License

```
* zlib.h -- interface of the 'zlib' general purpose compression library version 1.2.3,
July 18th, 2005
Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler
This software is provided 'as-is', without any express or implied warranty.  In no
event will the authors be held liable for any damages arising from the use of this
software.
Permission is granted to anyone to use this software for any purpose,including
commercial applications, and to alter it and redistribute it freely, subject to the
following restrictions:
  1. The origin of this software must not be misrepresented; you must not claim that
you wrote the original software. If you use this software in a product, an
acknowledgment in the product documentation would be appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.
  Jean-loup Gailly jloup@gzip.org
  Mark Adler madler@alumni.caltech.edu
```

A.4 General BSD License

```
Copyright (c) 1998, Regents of the University of California
All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
"Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.
"Redistributions in binary form must reproduce the above copyright notice, this list of
conditions and the following disclaimer in the documentation and/or other materials
provided with the distribution.
"Neither the name of the <ORGANIZATION> nor the names of its contributors may be used
to endorse or promote products derived from this software without specific prior
written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

A.5 General MIT License

Copyright (c) 1998, Regents of the Massachusetts Institute of Technology
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

A.6 Unicode License

UNICODE, INC. LICENSE AGREEMENT - DATA FILES AND SOFTWARE
Unicode Data Files include all data files under the directories http://www.unicode.org/Public/, http://www.unicode.org/reports/, and http://www.unicode.org/cldr/data/ . Unicode Software includes any source code published in the Unicode Standard or under the directories http://www.unicode.org/Public/, http://www.unicode.org/reports/, and http://www.unicode.org/cldr/data/.
NOTICE TO USER: Carefully read the following legal agreement. BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"), YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.
COPYRIGHT AND PERMISSION NOTICE
Copyright 1991-2006 Unicode, Inc. All rights reserved. Distributed under the Terms of Use in http://www.unicode.org/copyright.html.
Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that (a) the above copyright notice(s) and this permission notice appear with all copies of the Data Files or Software, (b) both the above copyright notice(s) and this permission notice appear in associated documentation, and (c) there is clear notice in each modified Data File or in the Software as well as in the documentation associated with the Data File(s) or Software that the data or software has been modified.
THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.
Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.
_____Unicode and the Unicode logo are trademarks of Unicode, Inc., and may be registered in some jurisdictions. All other trademarks and registered trademarks mentioned herein are the property of their respective owners

A.7 その他の帰属

Adobe, Acrobat, and the Acrobat Logo are registered trademarks of Adobe Systems Incorporated.
FAST Instream is a trademark of Fast Search and Transfer ASA.
HP-UX is a registered trademark of Hewlett-Packard Company.
IBM, Informix, and DB2 are registered trademarks of IBM Corporation.
Jaws PDF Library is a registered trademark of Global Graphics Software Ltd.
Kofax is a registered trademark, and Ascent and Ascent Capture are trademarks of Kofax Image Products.
Linux is a registered trademark of Linus Torvalds.
Mac is a registered trademark, and Safari is a trademark of Apple Computer, Inc.
Microsoft, Windows, and Internet Explorer are registered trademarks of Microsoft Corporation.
MrSID is property of LizardTech, Inc. It is protected by U.S. Patent No. 5,710,835. Foreign Patents Pending.
Oracle is a registered trademark of Oracle Corporation.
Portions Copyright 1994-1997 LEAD Technologies, Inc. All rights reserved.
Portions Copyright 1990-1998 Handmade Software, Inc. All rights reserved.
Portions Copyright 1988, 1997 Aladdin Enterprises. All rights reserved.
Portions Copyright 1997 Soft Horizons. All rights reserved.
Portions Copyright 1995-1999 LizardTech, Inc. All rights reserved.
Red Hat is a registered trademark of Red Hat, Inc.
Sun is a registered trademark, and Sun ONE, Solaris, iPlanet and Java are trademarks of Sun Microsystems, Inc.
Sybase is a registered trademark of Sybase, Inc.
UNIX is a registered trademark of The Open Group.
Verity is a registered trademark of Autonomy Corporation plc

索引

記号

A

C

G

H

I

J

M

N

P

R

S

T

U

W

X

い

え

か

き

く

こ

さ

し

す

せ

た

て

と

な

ね

の

は

ふ

ま

め

ゆ

よ

ら

り

れ

わ