BEA WebLogic Server™

Administration Console の拡張

BEA WebLogic Server バージョン 7.0
マニュアルの日付 :2002 年 6 月
改訂 : 2002 年 6 月 28 日

著作権

限定的権利条項

商標または登録商標

Administration Console の拡張

パート番号	マニュアルの日付	ソフトウェアのバージョン
なし	2002 年 6 月 28 日	WebLogic Server バージョン 7.0

目次

このマニュアルの内容

1. Administration Console の拡張

2. コンソール拡張タグ ライブラリの使い方

3. コンソール拡張でのローカライゼーションの使い方

このマニュアルの内容

このマニュアルでは、WebLogic Server Administration Console のカスタム拡張を作成する方法について説明します。

このマニュアルの内容は以下のとおりです。

- 第 1 章「Administration Console の拡張」では、Administration Console 拡張を作成する手順について説明します。
- 第 2 章「コンソール拡張タグ ライブラリの使い方」では、Administration Console 拡張の作成に使用する JSP タグ ライブラリについて説明します。
- 第 3 章「コンソール拡張でのローカライゼーションの使い方」では、Administration Console 拡張でローカライゼーションを使用する方法について説明します。

対象読者

このマニュアルは主に、WebLogic Server Administration Console の拡張を作成するアプリケーション開発者を対象としています。WebLogic Server プラットフォーム、Java プログラミング、WebLogic Server Mbean、XML、および Java Server Pages に、読者が精通していることを前提として書かれています。

e-docs Web サイト

BEA 製品のドキュメントは、BEA の Web サイトで入手できます。BEA のホーム ページで [製品のドキュメント] をクリックするか、または「e-docs」という製品ドキュメント ページ（http://edocs.beasys.co.jp/e-docs/）を直接表示してください。

このマニュアルの印刷方法

Web ブラウザの [ファイル | 印刷] オプションを使用すると、Web ブラウザからこのマニュアルを一度に 1 ファイルずつ印刷できます。

このマニュアルの PDF 版は、Web サイトで入手できます。PDF を Adobe Acrobat Reader で開くと、マニュアルの全体（または一部分）を書籍の形式で印刷できます。PDF を表示するには、WebLogic Server ドキュメントのホーム ページを開き、[ドキュメントのダウンロード] をクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader がない場合は、Adobe の Web サイト（http://www.adobe.co.jp/）で無料で入手できます。

関連情報

以下の WebLogic Server ドキュメントには、Administration Console 拡張の作成に関連する情報が含まれています。

- 『管理者ガイド』（http://edocs.beasys.co.jp/e-docs/wls/docs70/adminguide/index.html）
- 『WebLogic JSP プログラマーズ ガイド』（http://edocs.beasys.co.jp/e-docs/wls/docs70/jsp/index.html）
- Sun Microsystems, Inc. の Java サイト（http://java.sun.com/）

サポート情報

BEA WebLogic Server のドキュメントに関するユーザからのフィードバックは弊社にとって非常に重要です。質問や意見などがあれば、電子メールで **docsupport-jp@beasys.com** までお送りください。寄せられた意見については、WebLogic Server のドキュメントを作成および改訂する BEA の専門の担当者が直に目を通します。

電子メールのメッセージには、BEA WebLogic Server 7.0 リリースのドキュメントをご使用の旨をお書き添えください。

本バージョンの BEA WebLogic Server について不明な点がある場合、または BEA WebLogic Server のインストールおよび動作に問題がある場合は、BEA WebSupport（**www.bea.com**）を通じて BEA カスタマ サポートまでお問い合わせください。カスタマ サポートへの連絡方法については、製品パッケージに同梱されているカスタマ サポート カードにも記載されています。

カスタマ サポートでは以下の情報をお尋ねしますので、お問い合わせの際はあらかじめご用意ください。

- お名前、電子メール アドレス、電話番号、ファクス番号
- 会社の名前と住所
- お使いの機種とコード番号
- 製品の名前とバージョン
- 問題の状況と表示されるエラー メッセージの内容

表記規則

このマニュアルでは、全体を通して以下の表記規則が使用されています。

表記法	適用
太字	用語集で定義されている用語を示す。
〔Ctrl〕+〔Tab〕	複数のキーを同時に押すことを示す。
斜体	強調または書籍のタイトルを示す。

等幅テキスト	コード サンプル、コマンドとそのオプション、データ構造体とそのメンバー、データ型、ディレクトリ、およびファイル名とその拡張子を示す。等幅テキストはキーボードから入力するテキストも示す。
	例:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

太字の等幅テキスト	コード内の変数を示す。 例: `void **commit** ( )`
斜体の等幅テキスト	コード内の変数を示す。 例: `String *expr*`
すべて大文字のテキスト	デバイス名、環境変数、および論理演算子を示す。 例: LPT1 SIGNON OR
{ }	構文の中で複数の選択肢を示す。実際には、この括弧は入力しない。

表記法	適用
[ ]	構文の中で任意指定の項目を示す。実際には、この括弧は入力しない。 例： `buildobjclient [-v] [-o name ] [-f file-list]...[-l file-list]...`
|	構文の中で相互に排他的な選択肢を区切る。実際には、この記号は入力しない。
...	コマンドラインで以下のいずれかを示す。 ■ 引数を複数回繰り返すことができる。 ■ 任意指定の引数が省略されている。 ■ パラメータや値などの情報を追加入力できる。 実際には、この省略符号は入力しない。 例： `buildobjclient [-v] [-o name ] [-f file-list]...[-l file-list]...`
. . .	コード サンプルまたは構文で項目が省略されていることを示す。 実際には、この省略符号は入力しない。

1 Administration Console の拡張

このマニュアルでは、BEA WebLogic Server Administration Console を拡張する方法について説明します。Administration Console を拡張すると、標準コンソール ページと一緒に表示する独自のコンソール画面を作成できます。以下の節では、コンソールを拡張するための手順について説明します。

Administration Console の拡張の概要

BEA WebLogic Server Administration Console は、WebLogic Server ドメインを管理するためのブラウザ ベースのグラフィカル ユーザ インタフェースです。Administration Console の詳細については、『Administration Console オンライン ヘルプ』の「Administration Console について」を参照してください。

Administration Console を拡張するには、標準の Administration Console 画面と一緒に表示する画面とナビゲーション要素を追加します。

コンソール拡張は、標準の Administration Console に組み込まれていない機能、または既存の機能の代替インタフェースを提供します。たとえば、コンソール拡張を使用すると、以下のことができます。

- WebLogic Server にデプロイするアプリケーションを独自に管理する
- サードパーティ システムを管理する
- カスタム セキュリティ プロバイダを管理する (詳細については、「カスタム セキュリティ プロバイダ用のコンソール拡張の記述」を参照)
- WebLogic Server ドメイン向けにカスタマイズしたモニタおよび管理画面を作成する

Administration Console 拡張を作成するには、Java プログラミング、JavaServer Pages (JSP)、HTML、および WebLogic Server Mbean に関する中級レベルの知識が必要です。Mbean は、WebLogic Server ドメインのシステム管理に使用する Java オブジェクトです。WebLogic MBean と WebLogic Server システム管理インフラストラクチャの詳細については、『管理者ガイド』の「システム管理のインフラストラクチャ」を参照してください。

Administration Console 拡張のサンプル コードは、BEA dev2dev Web サイトから入手できます。Sample Administration Console Extension (WLS 7.0) をクリックし、`ConsoleExtensionExample.zip` ファイルをダウンロードしてください。独自のコンソール拡張を作成するには、このサンプルのファイルが必要です。

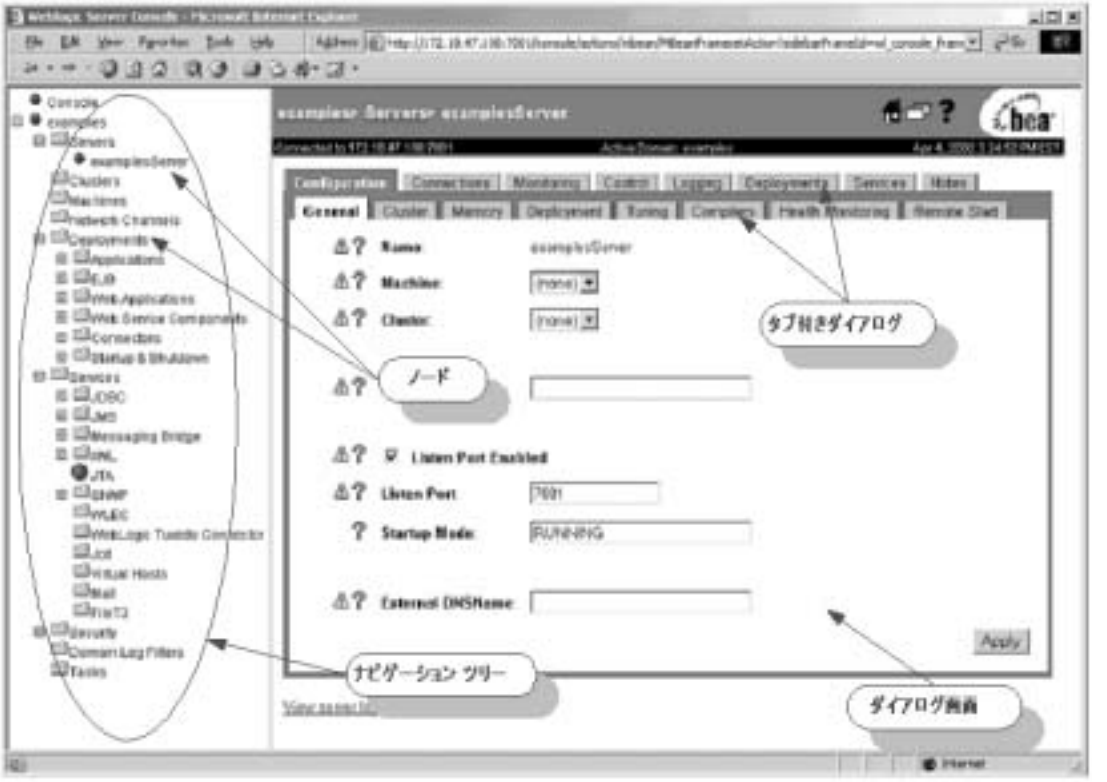
Administration Console 拡張のビジュアル要素

Administration Console 拡張には、以下のビジュアル要素を組み込むことができます (図 1-1 を参照)。

- ナビゲーション ツリー。ナビゲーション ツリーは、コンソールのダイアログ画面間を移動するための Java アプレットです。
- ノード。ノードは、ナビゲーション ツリーのブランチです。ノードは別のノードを含むか、または Administration Console の右ペインに表示されるダイアログ画面を呼び出すことができます。コンソール拡張では、ナビゲーション ツリーに 1 つまたは複数のノードを追加できます。

- タブ付きダイアログ。最大 2 レベルのタブ付きダイアログを作成できます。コンソール拡張画面はダイアログの一部です。
- ダイアログ画面。ダイアログ画面は、タブ付きダイアログの 1 つを選択すると表示されます。ダイアログ画面には、コンソール拡張の機能が表示されます。

図 1-1　Administration Console のビジュアル要素



コンソール拡張のプログラマティック要素

コンソール拡張を作成するには、以下のプログラマティック要素を作成します。

- この節で説明する要素、および拡張コンソール画面を実装するために必要な Java クラスまたは JSP タグ ライブラリを収めた Web アプリケーション。

- コンソール拡張へのリンクが表示される Administration Console ナビゲーション ツリー内の新しいノードを定義する Java クラス。このクラスは、コンソール拡張に必要な機能の初期化にも使用できます。このクラスは、WebLogic API の一部であるインタフェースを実装します。このクラスは、特別なコンテキスト パラメータを使用して Web アプリケーションに登録します。
- ナビゲーション ツリー内の新しいノードの動作を定義し、新しいノードの下に表示される子ノードを (任意に) 定義する JavaServer Page (JSP)。ノードを右クリックしたときに表示されるメニュー オプションも定義できます。
- コンソール ダイアログ画面を定義する 1 つまたは複数の JSP。標準の Administration Console 画面と類似のタブ付きインタフェースを作成するための JSP タグ ライブラリが用意されています。標準のコンソール ページと同じルック アンド フィールの画面を作成するための標準 HTML スタイル シートを利用できます。
- (省略可能) コンソール拡張に表示されるテキストとラベルの文字列を参照するためのローカライゼーション カタログ。ローカライゼーション カタログは XML を使用して作成します。

Administration Console 拡張の主要な作成手順

Administration Console 拡張を作成するには、次の手順が必要です。

1. Administration Console 拡張を定義する Java クラスを作成します。このクラスでは、コンソール拡張をナビゲーション ツリー内のどこに表示するかを定義し、拡張で必要とされる追加機能を定義します。1-5 ページの「NavTreeExtension インタフェースの実装」を参照してください。
2. ナビゲーション ツリーの動作を定義します。この手順では、手順 1 で定義したノードの下に表示する複数のノードを定義できます。また、右クリック メニューとアクションを定義できます。1-8 ページの「ナビゲーション ツリーの設定」を参照してください。

3. コンソール拡張画面に表示する JSP を記述します。ローカライゼーション カタログ内の文字列を参照することによって、ローカライズされたテキストを使用できます。標準のタグ ライブラリを使用すると、標準の Administration Console と同じようなタブ付きダイアログ画面を作成し、ローカライゼーション カタログにアクセスできます。1-11 ページの「コンソール画面 JSP の記述」を参照してください。
4. JSP、カタログ、および Java クラスを Web アプリケーションとしてパッケージ化します。1-15 ページの「Administration Console 拡張のパッケージ化」を参照してください。
5. コンソール拡張が収められた Web アプリケーションを対象 WebLogic Server ドメインの管理サーバにデプロイします。1-18 ページの「Administration Console 拡張のデプロイ」を参照してください。

NavTreeExtension インタフェースの実装

コンソール拡張を定義するには、`weblogic.management.console.extensibility.Extension` を拡張し、`weblogic.management.console.extensibility.NavTreeExtension` を実装する Java クラスを記述します。この Java クラスのサンプルについては、1-19 ページの「NavTreeExtension インタフェースを実装するためのサンプル Java クラス」を参照してください。

注意： カスタム セキュリティ プロバイダのコンソール拡張を作成する場合は、`weblogic.management.console.extensibility.NavTreeExtension` インタフェースではなく、`weblogic.management.console.extensibility.SecurityExtension` インタフェースを実装します (詳細については、「カスタム セキュリティ プロバイダ用のコンソール拡張の記述」を参照)。

Java クラスを記述するには、次の手順に従います。

1. コンソール拡張をナビゲーション ツリーのどこに (どのノードの下に) 表示するかを決定します。コンソールの各ノードは MBean オブジェクトに関連付けられます。この手順を使用してこれらの MBean オブジェクトの 1 つにコン

ソール拡張を関連付けることによって、そのコンソール拡張はいずれかの既存ノードの下に新ノードとして表示されます（MBean は WebLogic Server ドメインのコンフィグレーションに使用する Java オブジェクト）。

コンソール拡張を表すノードをどこに配置するかは、そのコンソール拡張の機能によって決める必要があります。たとえば、コンソール拡張がドメインの WebLogic Server インスタンスに関連している場合、そのコンソール拡張を `ServerMBean` (`weblogic.management.configuration.ServerMBean`) に関連付けて、そのコンソール拡張ノードを [サーバ] ノードに配置します。

コンソール拡張は、ノードの下に表示されるコンフィグレーション済みオブジェクトの各インスタンスの下に表示されます。たとえば、[サーバ] ノード (`ServerMBean`) を選択した場合、コンソール拡張はドメイン内の各コンフィグレーション済みサーバの下に表示されます。MBean のリストについては、`weblogic.management.congfiguration` パッケージの Javadoc を参照してください。

コンソール拡張をナビゲーション ツリーの最上位 (ドメイン) レベルに表示させる場合、その拡張を `DomainMBean` (`weblogic.management.configuration.DomainMBean`) に関連付けます。コンソールにはドメインの 1 つのインスタンスしか表示されないので、コンソール拡張は一度しか表示されません。

2. コンソール拡張に関連付けられる MBean クラスの `import` 文を追加します。コンソール拡張にアクセスするためのナビゲーション ツリーは、この MBean のノードの子として表示されます。次に例を示します。

   ```
   import weblogic.management.configuration.DomainMBean.
   ```

3. 次の `import` 文を追加します。

   ```
   import weblogic.management.console.extensibility.
   NavTreeExtension;
   ```

4. コンソール拡張の機能で必要となる場合、次の `import` 文を追加できます。

   ```
   import weblogic.management.console.extensibility.Catalog;
   import weblogic.management.console.extensibility.Extension;
   import javax.servlet.jsp.PageContext;
   ```

5. このクラスのクラス名を宣言します。次に例を示します。

   ```
   final public class ExampleConsoleExtension extends Extension
   implements NavTreeExtension
   ```

6. パブリック コンストラクタを引数なしで追加します。次に例を示します。

```
public ExampleConsoleExtension() {}
```

7. `getNavExtensionFor()` メソッドを定義します。Administration Console は自身を初期化するときにこのメソッドを呼び出し、ナビゲーション ツリーの各ノードを作成するときに関連する MBean の名前を `Object` 引数として渡します。

   このメソッドでは、`Object` 引数がコンソール拡張を表示するノードに関連付けられる MBean のインスタンスであるかどうかをテストします。`Object` がこの MBean のインスタンスである場合、このメソッドはナビゲーション ツリー内のノードの動作を定義する JSP ページの URL を返します。それ以外の場合、`null` を返します。次に例を示します。

   ```
   public String getNavExtensionFor(Object key)
   {
     if (key instanceof DomainMBean) {
        System.out.println(
              "\nFound an instance of the DomainMbean\n");
        return "domain_navlink.jsp";
     }
     return null;
   }
   ```

   上記の例では、Administration Console は `DomainMBean` のノードを作成するときにこのメソッドを実行し、ドメインの名前を `Object` 引数として渡します。`Object` は `DomainMBean` のインスタンスであるため、このメソッドは URL `domain_navlink.jsp` を返します。

   注意： カスタム セキュリティ プロバイダのコンソール拡張を作成する場合は、`getNavExtensionFor()` メソッドを定義しないでください。代わりに、「SecurityExtension インタフェースの使用によるカスタム セキュリティ プロバイダ関連の Administration Console ダイアログ画面の置き換え」で説明されているメソッドのいずれかを定義します。詳細については、SecurityExtension インタフェースの Javadoc を参照してください。

   `System.out.println` 文は省略可能で、メッセージを標準出力に表示する場合にのみ使用します。

8. 場合によっては、`weblogic.managment.console.extensions.Extension` クラスのメソッドを呼び出してコンソール拡張の機能を実装する必要があります。必要な使い方については、このマニュアルの範囲を超えています。詳細については、`weblogic.management.console.extensibility` パッケージの Javadoc を参照してください。

9. クラスをコンパイルして、コンソール拡張が格納されている Web アプリケーションの `WEB-INF/classes` ディレクトリにそのクラスを表示します。クラスをコンパイルするには、開発環境を設定して WebLogic Server クラスを組み込みます。詳細については、「開発環境の構築」を参照してください。

ナビゲーション ツリーの設定

1-5 ページの「NavTreeExtension インタフェースの実装」で説明したとおりに記述した Java クラスの `getNavExtensionFor()` メソッドは、ナビゲーション ツリーに表示される各コンソール拡張に対応する URL を返します。この URL は、このノードの動作を定義する JSP を指し示します。この JSP では、以下のものを定義します。

- ノードの子として表示されるサブノード。
- ノードのラベルの左側に表示されるアイコン。
- 右クリック メニュー。メニューのアイテムはコンソールに表示される URL を呼び出すことができます。また、右クリック メニューに表示される区切り線 (メニュー リスト内の水平線) を定義できます。

ナビゲーション ツリー ノードを定義する JSP を定義するには、次の手順に従います。

1. `getNavExtensionFor()` メソッドから返される URL と一致する名前で新しい JSP ファイルを作成します (`domain_navlink.jsp` など)。
2. 作成した JSP ファイルを、コンソール拡張が収められている Web アプリケーションの最上位ディレクトリに保存します。
3. 次の taglib 文を追加します。

   ```
   <%@ taglib uri='console_extension_taglib.tld' prefix='wl' %>
   ```

4. (省略可能) この JSP のオブジェクトにアクセスする必要がある場合、次の JSP タグを追加します。

   ```
   <wl:extensibility-key
   id='domainKey'
   class='MyObjectClass' />
   ```

   `MyObjectClass` は、アクセスするオブジェクトの Java クラス名です。

詳細については、2-8 ページの「<wl:extensibility-key> タグ」を参照してください。

5. 1 つまたは複数の `<wl:node>` タグを追加します。これらのタグでは、ナビゲーション ツリーに表示されるノードを記述します。`<wl:node>` タグをネストして子ノードを作成できます。`<wl:node>` タグの `url`、`label`、`labelId`、`icon`、`expanded`、`target`、および `font` 属性を使用すると、このノードの外観と機能を定義できます。これらの属性の詳細については、2-2 ページの「<wl:node> タグ」を参照してください。

   label 属性では、タブの表示名を定義します。この名前をローカライズする場合、labelId 属性を使用してローカライゼーション ライブラリ内でこの名前を参照します。ローカライゼーションの詳細については、3-1 ページの「コンソール拡張でのローカライゼーションの使い方」を参照してください。

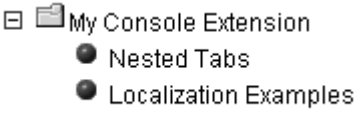
   icon 属性はイメージ ファイルを指し示し、そのイメージをナビゲーション ツリー内のこのノードのアイコンとして表示します。アイコンとして使用するイメージは、`extension_files/images` ディレクトリ内のサンプル アプリケーションに用意されています (「Administration Console 拡張のパッケージ化」の手順 1. を参照)。

   次に例を示します。

```
<wl:node
   label='<%="My Console Extension"%>'
   icon='/images/folder.gif'
   expanded='true'>

   <wl:node
      label='Nested Tabs'
      icon='/images/bullet.gif'
      url='/dialog_domain_example.jsp'>
   </wl:node>

   <wl:node label='Localization Examples'
      icon='/images/bullet.gif'>
   </wl:node>

</wl:node>
```

   上のコードによって、図 1-2 に示すナビゲーション ツリー ノードが作成されます。

図 1-2 ナビゲーション ツリー ノード

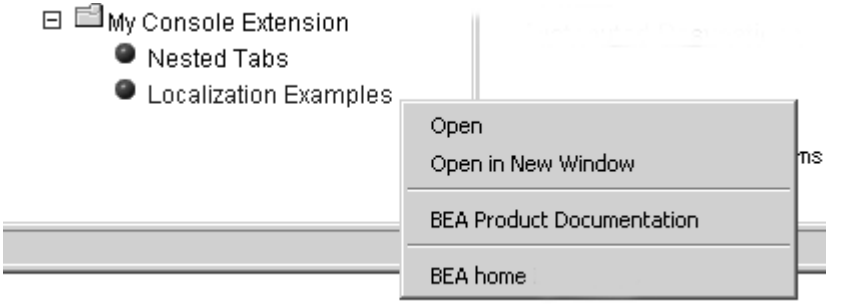


6. (省略可能) 1 つまたは複数の <wl:menu> タグを追加して、右クリック メニュー オプションを作成します。<wl:menu> タグの属性は、label、labelId、url、および target です。詳細については、2-4 ページの「<wl:menu> および <wl:menu-separator> タグ」を参照してください。たとえば、前の手順の例で定義した Localization Examples ノードに <wl:menu> タグを追加するには、次のコードを使用します。

```
...

   <wl:node
      label='Localization Examples'
      icon='/images/bullet.gif'>
      <wl:menu
         label='BEA Product Documentation'
         url='http://e-docs.bea.com/index.html'
         target='_blank'/>
      <wl:menu-separator/>1

       <wl:menu
           label='BEA home'
           url='http://www.bea.com'
            target='_blank'/>

   </wl:node>

...
```

上のコードによって、図 1-3 に示す右クリック メニューが作成されます。

図 1-3 ナビゲーション ノードと右クリック メニュー



コンソール画面 JSP の記述

コンソール拡張によって生成される実際のダイアログ画面は、ユーザがナビゲーション ツリー内のコンソール拡張のノードをクリックしたときに Administration Console の右ペインに表示されます。これらの画面を作成するには、付属の JSP タグ ライブラリを使用して JSP を記述します。タグ ライブラリのリファレンス情報については、2-1 ページの「コンソール拡張タグ ライブラリの使い方」を参照してください。表示される JSP は、「ナビゲーション ツリーの設定」で作成した JSP の `<wl:node>` タグの `url` 属性によって決定されます。

ダイアログ画面は、`<wl:tab>` JSP タグで定義する 1 つまたは複数のタブ付きダイアログを使用して作成します。これらのタブにはサブ タブをネストできますが、ネストは 1 レベルしかできません。各タブには、テキスト ラベルを明示的に指定するか、またはローカライゼーション カタログ内でそのタブのローカライズされたラベルを参照するためのラベル ID を指定できます。

各タブ (`<wl:tab>...</wl:tab>` タグ) の中では、JSP および HTML コードを使用してコンソール拡張の機能を作成できます。これらの中に表示されるテキストも、ローカライゼーション カタログを参照することによってローカライズできます。ローカライゼーション (コンソール拡張を複数の言語で表示する機能) の使い方については、3-1 ページの「コンソール拡張でのローカライゼーションの使い方」を参照してください。

注意： コンソール拡張のユーザ インタフェースを作成する際には、さまざまなプログラミング テクニックを使用できます。これらのテクニックについては、このマニュアルの範囲を超えています。詳細については、以下を参照してください。

- 『WebLogic JSP プログラマーズ ガイド』
- 『WebLogic JMX Service プログラマーズ ガイド』

次に、コンソール拡張画面を表示する基本的な JSP を作成する手順を示します。

1. この画面を呼び出す `<wl:node>` タグの `url` 属性で指定した URL と一致する JSP ファイルを新しく作成します (`domain_dialog.jsp` など)。
2. 作成した JSP ファイルを、コンソール拡張が収められている Web アプリケーションの最上位ディレクトリに保存します。
3. この JSP ファイルの最上位に次の `taglib` 文を挿入します。

   ```
   <%@ taglib uri='console_extension_taglib.tld' prefix='wl' %>
   ```

4. JSP ファイルに HTML および JSP ブロックを挿入します。コンソール拡張の表示は、HTML コードおよび HTML コードに変換される JSP コードによって定義されます。このため、表示コードを次の HTML タグ セットで囲みます。

   ```
   <html>
    <head>
      <wl:stylesheet/>
    </head>
    <body>
       <div class='content'>
           <wl:dialog>
              (Insert <wl:tab> statements here.)
           </wl:dialog>
       </div>
     </body>
   </html>
   ```

   `<head>...</head>` ブロック内の `<wl:stylesheet/>` タグは省略可能です。このタグを組み込むと、指定するテキストのフォーマットが標準 WebLogic Server Administration Console ページと一致します。

5. JSP ファイルに 1 つまたは複数の <wl:tab> タグを追加します (これらのタグを <wl:dialog>...</wl:dialog> タグの間に配置します)。各 <wl:tab> タグでは、Administration Console の右ペインに表示されるタブ付き画面を定義します。最上位タブの中には 1 つまたは複数のタブをネストできますが、サポートされるネスト レベルは 1 です。

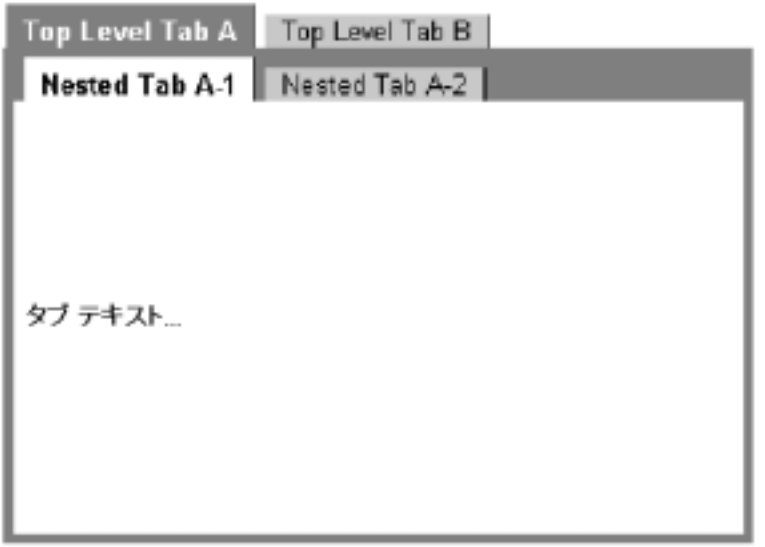
   各 <wl:tab> タグに対しては、name、label、および labelId 属性を定義できます。各 <wl:tab> タグには終了タグ (</wl:tab>) が必要です。これらの属性の使い方については、2-6 ページの「<wl:tab> タグ」を参照してください。<wl:tab> ブロックの中には、コンソール拡張ダイアログ画面の本体を表示する HTML および JSP コードを記述します。また、タブの表示ラベルをローカライズできます。詳細については、3-10 ページの「タブ ラベルのローカライズ」を参照してください。

   たとえば、次のコードでは 2 つの最上位タブが作成され、各タブには 2 つのタブがネストされます。これらのタブがコンソールでどのように表示されるかについては、図 1-4 を参照してください。

```
<wl:tab name='TopLevelTabA' label='Top Level Tab A'>

   <wl:tab name='NestedTabA1' label='Nested Tab A-1'>
       (Insert your JSP and/or HTML code
       for displaying your console extension here.)
   </wl:tab>

   <wl:tab name='NestedTabA2' label='Nested Tab A-2'>
      (Insert your JSP and/or HTML code
       for displaying your console extension here.)
   </wl:tab>

</wl:tab>

<wl:tab name='TopLevelTabB' label='Top Level Tab B'>

   <wl:tab name='NestedTabB1' label='Nested Tab B-1'>
       (Insert your JSP and/or HTML code
       for displaying your console extension here.)
   </wl:tab>

   <wl:tab name='NestedTabB2' label='Nested Tab B-2'>
      (Insert your JSP and/or HTML code
       for displaying your console extension here.)
   </wl:tab>

</wl:tab>
```

注意： この手順は、コンソール拡張を定義する基本的な JSP を作成するためのものです。ローカライゼーションなどの他の機能を実現するタグも存在しますが、ここでは示しません。これらのタグについては、別の節で説明します。

図 1-4 ネストされたタブ



Administration Console 拡張のローカライズ

前節の手順では、コンソール拡張を複数の言語で表示するためのローカライゼーションについては説明しませんでした。コンソール拡張をローカライズする手順には、特別な JSP タグの使用、ローカライゼーション カタログの記述、すべてのローカライゼーション カタログを表示する `index.xml` ファイルの記述などが含まれます。記述した `index.xml` ファイルとカタログ ファイルは、コンソール拡張を定義する Web アプリケーションにパッケージ化します (1-15 ページの「Administration Console 拡張のパッケージ化」を参照)。

ローカライゼーションの詳細については、3-1 ページの「コンソール拡張でのローカライゼーションの使い方」を参照してください。

Administration Console 拡張のパッケージ化

コンソール拡張の JSP と Java クラスは、J2EE Web アプリケーションとしてパッケージ化して WebLogic Server ドメインの管理サーバにデプロイします。

コンソール拡張をパッケージ化するには、次の手順に従います。

1. BEA dev2dev Web サイトから Sample Administration Console Extension をダウンロードします。Sample Administration Console Extension (WLS 7.0) をクリックし、`ConsoleExtensionExample.zip` ファイルをダウンロードしてください。独自のコンソール拡張を作成するには、このサンプルのファイルが必要です。
2. `ConsoleExtensionExample.zip` ファイルをハード ディスク上の一時ディレクトリに解凍します。このアーカイブの `extension_files` ディレクトリには、コンソール拡張に必要な以下のファイルが存在します。
   - `console_extension_taglib.tld`
   - `images/folder.gif`
     (ナビゲーション ツリーでこれらのアイコンを使用する場合にのみ必要)
   - `images/bullet.gif`
     (ナビゲーション ツリーでこれらのアイコンを使用する場合にのみ必要)
   - `deployment_descriptor_templates/web.xml`
     (コンソール拡張用に修正可能なテンプレート)
   - `deployment_descriptor_templates/weblogic.xml`
     (コンソール拡張用に修正可能なテンプレート)

   上記のファイルを、このコンソール拡張パッケージ化手順で示す場所にコピーします。図 1-5 に、これらのファイルの Web アプリケーション内での格納場所を示します。
3. Web アプリケーションの `web.xml` デプロイメント記述子を記述します (`ConsoleExtensionExample.zip` ファイルのテンプレートから作成できます)。`web.xml` デプロイメント記述子には、コンソール拡張クラスの名前と JSP タグ ライブラリを記述します。デプロイメント記述子の作成には、テキスト エディタを使用するか、または WebLogic Server 配布キットの WebLogic Builder ツールを使用します。詳細については、『WebLogic Builder Online Help』を参照してください。

web.xml デプロイメント記述子には、次の例に示す要素が定義されている必要があります。「NavTreeExtension インタフェースの実装」の手順 5. で作成したクラスの名前を *MyConsoleExtension* に置き換えます。完全なパッケージ名を指定する必要があります。

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>
     My Weblogic Console Example Extension
  </display-name>
  <context-param>
    <param-name>weblogic.console.extension.class</param-name>
    <param-value>MyConsoleExtension</param-value>
  </context-param>

<taglib>
  <taglib-uri>console_extension_taglib.jar</taglib-uri>
  <taglib-location>
     WEB-INF/console_extension_taglib.tld
  </taglib-location>
</taglib>

</web-app>
```

また、コンソール拡張で必要とされる他の要素を追加できます。

4. weblogic.xml デプロイメント記述子を記述します (ConsoleExtensionExample.zip ファイルのテンプレートから作成できます)。weblogic.xml 記述子には、コンソール拡張が Administration Console 全体と同じセキュリティ コンテキストを共有するためのエントリを挿入します。デプロイメント記述子の作成には、テキスト エディタを使用するか、または WebLogic Server 配布キットの WebLogic Builder ツールを使用します。詳細については、『WebLogic Builder Online Help』を参照してください。

   weblogic.xml デプロイメント記述子には、次の例に示す要素を挿入する必要があります。

```
<!DOCTYPE weblogic-web-app PUBLIC "-//BEA
Systems, Inc.//DTD Web Application 7.0//EN"
"http://www.bea.com/servers/wls700/dtd/weblogic700-web-jar.dtd"
>

<weblogic-web-app>

   <session-descriptor>
```

```
      <session-param>
        <param-name>CookieName</param-name>
        <param-value>ADMINCONSOLESESSION</param-value>
      </session-param>
    </session-descriptor>

</weblogic-web-app>
```

また、コンソール拡張で必要とされる他の要素を追加できます。

5. console_extension_taglib.tld ファイルをサンプル アプリケーション (「Administration Console 拡張のパッケージ化」の手順 2. を参照) からコンソール拡張 Web アプリケーションの `WEB-INF` ディレクトリにコピーします。

6. 以下のディレクトリ構造例に示すとおり、コンソール拡張のコンポーネント (手順 1 および 2 で作成した `web.xml` および `weblogic.xml` デプロイメント記述子を含む) を配置し、コンソール拡張に必要な JSP、HTML、タグ ライブラリ記述子、Java クラス、またはイメージ ファイルを所定の場所に追加します。

図 1-5 コンソール拡張のサンプル ディレクトリ レイアウト

```
+---MyConsoleExtensionWebApp
|   |   dialog_domain_example.jsp
|   |   dialog_example.jsp
|   |   domain_navlink.jsp
|   |   server_navlink.jsp
|   |   *.html, *.jsp
|   |
|   +---images
|   |       bullet.gif
|   |       smiley.gif
|   |       *.gif, *.jpg
|   |
|   \---WEB-INF
|       |   console_extension_taglib.tld
|       |   *.tld
|       |   web.xml
|       |
|       +---catalogs
|       |       english.xml
|       |       german.xml
|       |       index.xml;
|       |       japanese.xml
|       |
|       \---classes
|                MyConsoleExtension.class
```

7. アプリケーションを .war アーカイブとしてパッケージ化します。たとえば、図 1-5 に示すディレクトリ レイアウトを使用して、MyConsoleExtensionWebApp ディレクトリにスイッチして次のコマンドを使用します。

   ```
   jar cvf MyConsoleExtension.war .
   ```

   Web アプリケーションは、.war アーカイブ ファイルを使用せずに展開ディレクトリ形式でデプロイできます。展開形式によるデプロイメントは、コンソール拡張の開発時に便利です。「Administration Console 拡張のデプロイ」を参照してください。

Administration Console 拡張のデプロイ

コンソール拡張が収められた Web アプリケーションを作成したら、その Web アプリケーションを対象 WebLogic Server ドメインの管理サーバにデプロイします。Web アプリケーションのデプロイについては、『WebLogic Server アプリケーションの開発』の「WebLogic Server デプロイメント」を参照してください。

コンソール拡張を有効にするには、Web アプリケーションをデプロイした後に管理サーバを再起動する必要があります。ナビゲーション ツリー ノードを定義する JSP を修正した場合、管理サーバを再起動してその変更をナビゲーション ツリーに反映させる必要があります。

Administration Console の再起動後、ダイアログ画面を定義する JSP を変更した場合、Web アプリケーションを再デプロイしてその変更を反映させます。Web アプリケーションを再デプロイするには、Administration Console を使用します。

1. ナビゲーション ツリーで、[デプロイメント | Web アプリケーション] を選択します。
2. コンソール拡張の名前を選択します。
3. [デプロイ] タブをクリックします。
4. [再デプロイ] ボタンをクリックします。

NavTreeExtension インタフェースを実装するためのサンプル Java クラス

```
package weblogic.management.console.extensibility.example;


import javax.servlet.jsp.PageContext;
import weblogic.management.configuration.DomainMBean;
import weblogic.management.console.extensibility.Catalog;
import weblogic.management.console.extensibility.Extension;
import weblogic.management.console.extensibility.NavTreeExtension;

/**
 * <p> コンソール拡張のサンプル実装 </p>
 */
final public class ExampleConsoleExtension extends Extension
implements NavTreeExtension {

  // コンストラクタ

  /**
   * 引数を取らないパブリック  コンストラクタが必要
   */
  public ExampleConsoleExtension() {}

  // ============================================================
  // NavTreeExtension 実装

  public String getNavExtensionFor(Object key)
  {
      if (key instanceof DomainMBean) {
          System.out.println("==\n== Found instance of
DomainMbean\n==");
          return "domain_navlink.jsp";
      }
          return null;
  }

  // ============================================================
  // オプションの拡張メソッド

   /**
   * <p> この例では特別な初期化作業は必要ない
   * このため、このメソッドはメッセージを送信して
   * コンソールが拡張を見つけたことを知らせる </p>
   * <p> このメソッドを使用すると、コンソール拡張で必要な
   * 初期化を実行できる </p>
   */
  public void initialize()
```

```
  {
    System.out.println("==\n== Example Extension for Domain
Initialized!\n==");
  }

  /**
   * <p> ローカライゼーション カタログを参照することによって
   * 拡張の名前を返す </p>
   */
  public String getName(PageContext context)
  {
    return Catalog.Factory.getCatalog(context).
      getText("example.extension.name");
  }

  /**
   * <p> この拡張の簡単な説明を記述する </p>
   */
  public String getDescription(PageContext context)
  {
    return Catalog.Factory.getCatalog(context).
      getText("example.extension.description");
  }
}
```

2 コンソール拡張タグ ライブラリの使い方

以下の節では、コンソール拡張タグの属性について説明し、各属性の使用例を示します。

コンソール拡張 JSP タグ ライブラリの概要

WebLogic Server 配布キットには、コンソール拡張を作成するときに使用できる JSP タグ ライブラリが用意されています。タグ ライブラリを使用すると、以下のことを行うことができます。

- Administration Console ナビゲーション ツリーに新しいノードを作成する
- Administration Console ナビゲーション ツリー内のノード用の右クリック オプションを作成する
- コンソール拡張を表示するためのタブ付きインタフェースを作成する
- ナビゲーション ツリーとコンソール拡張ツリーに表示されるテキストをローカライズする（テキストを別の言語で表現する）

タグ ライブラリ属性のリファレンス

以下の節では、コンソール拡張タグ ライブラリの各タグの使い方について説明します。

<wl:node> タグ

<wl:node> タグは、Administration Console のナビゲーション ツリーに新しいノードを作成するために使用します。

次に、<wl:node> タグの使用例を示します。

コード リスト 2-1　<wl:node> タグの使用例

```
<wl:node
   label='<%="My Console Extension"%>'
   icon='/images/smiley.gif'
   expanded='true'>
```

```
   <wl:node
      label='My 1st nested node'
      icon='/images/bullet.gif'
      url='/dialog_domain_example.jsp'>
   </wl:node>

   <wl:node label='My 2nd nested node'
      icon='/images/bullet.gif'>
   </wl:node>

</wl:node>
```

表 2-1 <wl:node> タグの属性

属性	説明
icon	ナビゲーション ツリーに表示されるこのノードのイメージ ファイル（.gif または .jpg）の URL。 URL は、絶対 URL（http://somesite.com/images/myIcon.gif など）か、またはコンソール拡張が収められている Web アプリケーションを起点とする相対 URL（/images/myIcon.gif など）。
label	このノード用に表示するテキスト ラベル。labelId 属性を定義する場合、この属性は使用しないこと。
labelId	このノードのローカライズされたテキスト ラベルのカタログ ID。label 属性を定義する場合、この属性は使用しないこと。
url	ユーザがこのノードをクリックしたときに Administration Console に表示されるページの URL。 URL は、絶対 URL（http://somesite.com/myPage.jsp など）か、またはコンソール拡張が収められている Web アプリケーションを起点とする相対 URL。

属性	説明
target	url 属性で指定した URL を表示するブラウザ フレームの名前。この属性を指定しない場合、URL は Administration Console の右ペインに表示される。任意の名前を選択するか、または以下のキーワードの 1 つを使用できる。 ■ _top - コンソールが表示されているブラウザ ウィンドウに、そのコンソールに代わって URL を表示する ■ _blank - url 属性で指定されたページを新しいブラウザ ウィンドウに表示する
expanded	true または false に設定する。true に設定した場合、ノードが展開された状態で表示される（すべての子ノードが表示される）。 デフォルトは false。
font	ノードのテキスト ラベルを表示するために使用するフォント。使用可能なフォントはブラウザに依存する。

<wl:menu> および <wl:menu-separator> タグ

<wl:menu> は、<wl:node> タグで定義したナビゲーション ツリー内のノードを右クリックしてアクセスするメニューとアクションを作成するために使用します。<wl:menu-separator> タグは、右クリック メニューに区切り線を挿入するために使用します。

コード リスト 2-2　<wl:menu> および <wl:menu-separator> タグの使用例

```
...

   <wl:node
      label='My 2nd nested node'
      icon='/images/bullet.gif'>
      <wl:menu
         label='BEA Product Documentation'
         url='http://e-docs.bea.com/index.html'
         target='_blank'/>
```

```
      <wl:menu-separator/>

       <wl:menu
           label='BEA home page'
           url='http://www.bea.com'
            target='_blank'/>

   </wl:node>

...
```

上のコードによって、ナビゲーション ツリー内の「My 2nd Nested Node」の下に右クリック メニューが作成されます。

表 2-2 <wl:menu> タグの属性

属性	説明
label	このメニュー アイテム用に表示されるテキスト ラベル。labelId 属性を定義する場合、この属性は使用しないこと。
labelId	このメニュー アイテムのローカライズされたテキスト ラベルのカタログ ID。label 属性を定義する場合、この属性は使用しないこと。
url	コンソールに表示されるページの絶対 URL または Web アプリケーション ルートからの相対 URL。
target	url 属性で指定した URL を表示するブラウザ フレームの名前。この属性を指定しない場合、URL は Administration Console の右ペインに表示される。任意の名前を選択するか、または以下のキーワードの 1 つを使用できる。 _top - コンソールが表示されているブラウザ ウィンドウに、そのコンソールに代わって URL を表示する _blank - url 属性で指定されたページを新しいブラウザ ウィンドウに表示する

<wl:menu-separator> タグには属性はありません。

<wl:tab> タグ

<wl:tab> タグは、コンソール拡張にタブ付きインタフェースを作成するために使用します。<wl:tab> タグを別の <wl:tab> タグの中にネストすると、ネストされたタブ付き画面を作成できます。サポートされるネスト レベルは 1 です。

次に、<wl:tab> タグの使用例を示します。

コード リスト 2-3　<wl:tab> タグの使用例

```
<wl:tab name='TopLevelTabA' label='Top Level Tab A'>

   <wl:tab name='NestedTabA1' label='Nested Tab A-1'>
       (Insert your JSP and/or HTML code
       for displaying your console extension here.)
   </wl:tab>

   <wl:tab name='NestedTabA2' label='Nested Tab A-2'>
      (Insert your JSP and/or HTML code
       for displaying your console extension here.)
   </wl:tab>

</wl:tab>

<wl:tab name='TopLevelTabB' label='Top Level Tab B'>

   <wl:tab name='NestedTabB1' label='Nested Tab B-1'>
       (Insert your JSP and/or HTML code
       for displaying your console extension here.)
   </wl:tab>

   <wl:tab name='NestedTabB2' label='Nested Tab B-2'>
      (Insert your JSP and/or HTML code
       for displaying your console extension here.)
   </wl:tab>

</wl:tab>
```

表 2-3 <wl:tab> タグの属性

属性	説明
name	タブの名前。この名前の中にピリオド（.）を使わないこと。labelId 属性を指定しない場合、コンソールはローカライゼーション カタログから次の形式のエントリを検索する。 tab + *name* たとえば、*name* を config に設定した場合、コンソールは ID tab.config を使用してカタログからローカライズ テキストを検索してタブのラベルを決定する。
label	タブのタイトルとして表示されるテキスト。labelId 属性を定義する場合、この属性は定義しないこと。この属性は、ローカライゼーション カタログを使用しない場合にのみ使用する。
labelId	タブのローカライズ ラベルのカタログ ID（name 属性と異なる場合）。このテキストは、ローカライゼーション カタログ内で検索される。label 属性を定義する場合、この属性は定義しないこと。

<wl:dialog> タグ

<wl:dialog> タグは、タブ付きコンソール画面を定義する JSP のセクションを定義します。<wl:tab> タグは、<wl:dialog> ブロックの中に挿入する必要があります。

コード リスト 2-4 <wl:dialog> タグの使用例

```
...
<wl:dialog>
  <wl:tab>
     ....(Insert code for tabbed dialog screen here.)
  </wl:tab>
```

```
</wl:dialog>
...
```

<wl:dialog> タグには属性はありません。

<wl:stylesheet> タグ

<wl:stylesheet> タグを HTML <head> ブロックに挿入すると、コンソール画面に対して Administration Console と同じ表示スタイル（フォントや色など）が適用されます。

コード リスト 2-5　<wl:stylesheet> タグの使用例

```
<html>
 <head>
   <wl:stylesheet/>
 </head>
...
```

<wl:dialog> タグには属性はありません。

<wl:extensibility-key> タグ

<wl:extensibility-key> タグは、Java オブジェクトを表すスクリプト変数を作成します。このタグは、ナビゲーション ツリーを定義する JSP の中で使用できます。

注意： <wl:extensibility-key> タグは、コンソール ダイアログ画面を定義する JSP の中では使用できません。

コード リスト 2-6 <wl:extensibility-key> タグの使用例

```
<wl:extensibility-key
  id='domainKey'
  class='weblogic.management.configuration.DomainMBean' />

<%="Configuration Version is" +
domainKey.getConfigurationVersion()%>
```

属性	説明
id	スクリプト変数の名前。
class	スクリプト変数の Java クラス。

<wl:text> タグ

<wl:text> タグは、ローカライゼーション カタログのテキストを表示するために使用します。

コード リスト 2-7 <wl:text> タグの使用例

```
<wl:text textId='Text.3' textParamId='Param.1' />
<p>
<wl:text textId='Text.2' textParam="Blue"/>
```

表 2-4 <wl:text> タグの属性

属性	説明
style	（省略可能）テキストを表示するための HTML スタイル クラス。

属性	説明
text	表示する実際のテキスト。textID 属性を定義する場合、この属性は定義しないこと。
textId	表示するローカライズ テキストのカタログ ID。このテキストは、ローカライゼーション カタログ内で検索される。text 属性を定義する場合、この属性は定義しないこと。
textParamId	カタログから検索されたテキスト内の文字列 {0} に置き換わるテキストのローカライゼーション カタログ ID。textParam 属性を定義する場合、この属性は定義しないこと。
textParam	カタログから検索されたテキスト内の文字列 {0} に置き換わるリテラル文字列。textParamID 属性を定義する場合、この属性は定義しないこと。

コンソール拡張でのタグ ライブラリの使い方

コンソール拡張タグ ライブラリを使用するには、次の手順に従います。

1. 次のセクションを、コンソール拡張が収められている Web アプリケーションの web.xml デプロイメント記述子に追加します。

```
<taglib>
  <taglib-uri>console_extension_taglib.jar</taglib-uri>
  <taglib-location>
     WEB-INF/console_extension_taglib.tld
  </taglib-location>
</taglib>
```

2. console_extension_taglib.tld ファイルを、コンソール拡張が収められている Web アプリケーションの WEB-INF ディレクトリにコピーします。

3. コンソール拡張タグ ライブラリを使用する各 JSP の先頭に次の taglib 文を挿入します。

```
<%@ taglib uri='console_extension_taglib.jar' prefix='wl' %>
```

3　コンソール拡張でのローカライゼーションの使い方

以下の節では、Administration Console 拡張で使用されるテキストをローカライズする方法について説明します。

コンソール拡張のローカライゼーションの概要

Administration Console 拡張をローカライズすると、コンソール拡張をさまざまな言語で提供できます。ローカライズしたテキストは、提供する言語ごとに作成した特別なローカライゼーション カタログ ファイルに格納します。さらに、すべてのカタログ ファイルのリストを `index.xml` というファイルに記述します。作成した `index.xml` ファイルとカタログ ファイルは、コンソール拡張を定義する Web アプリケーションにパッケージ化します（1-15 ページの「Administration Console 拡張のパッケージ化」を参照）。

ローカライズ可能なコンソール拡張の部分は以下のとおりです。

- ナビゲーション ツリーのノードのラベル
- ナビゲーション ツリーの右クリック メニュー オプションのラベル

- コンソール画面のタブのラベル
- コンソール画面に表示するテキスト

コンソールで使用するローカライゼーション カタログの決定

コンソール アプリケーションは、Administration Console で設定された言語プリファレンス、ユーザの Web ブラウザで指定された言語と国、および index.xml ファイルの設定値を使用して、コンソールを表示するためのローカライゼーション カタログを決定します。index.xml ファイル（3-6 ページの「index.xml ファイルの記述」を参照）では、使用可能なすべてのローカライゼーション カタログのリストが定義され、それらのファイルと言語プリファレンス、および 1 つまたは複数のロケール設定が関連付けられます。ロケール設定には、国属性と言語属性があります。

コンソール アプリケーションは、Administration Console の [コンソール | プリファレンス] 画面で設定された言語プリファレンスをチェックします。言語が指定されている場合、コンソールは index.xml ファイルの <catalog> 要素の name 属性を参照して、その言語に関連付けられているローカライゼーション カタログを見つけます。

注意： [コンソール | プリファレンス] 画面で言語が明示的に指定されなかった場合、[言語] フィールドのデフォルトは英語です。プリファレンスが設定されていない場合、コンソールは index.xml ファイルの <locale> 要素を参照して、ユーザの Web ブラウザの設定値と一致する国と言語を見つけます。

コンソール拡張のローカライゼーションの主要手順

コンソール拡張のテキストをローカライズするには、次の手順に従います。

1. 提供する言語ごとに XML カタログ ファイルを作成します。これらの XML ファイルには、ローカライズされたテキストを参照するための ID 文字列を記述します。この ID 文字列は、各言語カタログで同じです。ID 文字列は、カタログから検索され、コンソールに表示されるテキストのブロックを指し示します。

   手順の詳細については、3-4 ページの「ローカライゼーション カタログの記述」を参照してください。

2. ローカライゼーション カタログのリストを記述した index.xml ファイルを作成します。

   手順の詳細については、3-6 ページの「index.xml ファイルの記述」を参照してください。

3. コンソール拡張タグ ライブラリの JSP タグを使用して、ローカライズしたテキストを表示します。各 JSP タグには、カタログ ID を指定するための属性が存在します。詳細については、3-9 ページの「JSP でのローカライズ テキストの使い方」を参照してください。

4. JSP、カタログ、および Java クラスを Web アプリケーションとしてパッケージ化します。詳細については、1-15 ページの「Administration Console 拡張のパッケージ化」を参照してください。

5. コンソール拡張が収められた Web アプリケーションを対象 WebLogic Server ドメインの管理サーバにデプロイします。詳細については、1-18 ページの「Administration Console 拡張のデプロイ」を参照してください。

ローカライゼーション カタログの記述

ローカライゼーション カタログは、XML 表記法を使用して記述します。XML エディタまたは通常のテキスト エディタを使用できます。サンプル カタログについては、3-8 ページの「サンプル ローカライゼーション カタログ」を参照してください。

ローカライゼーション カタログを記述するには、次の手順に従います。

1. ローカライゼーション カタログをプレーン テキスト ファイルとして作成し、コンソール拡張が格納されている Web アプリケーションの `WEB-INF/catalogs` ディレクトリに保存します。ローカライゼーション カタログのファイル名は、そのカタログを定義する `index.xml`（3-6 ページの「index.xml ファイルの記述」を参照）の `<catalog>` 要素の `file` 属性と一致している必要があります。

   カタログ ファイルには、以下の基本 XML 要素を記述する必要があります。

   ```
   <?xml version="1.0" ?>
   <catalog>

   (Insert your catalog data here.)

   </catalog>
   ```

2. ローカライゼーション テキスト エントリを作成します。
   - 1 つの単語または短い語句をローカライズするには、3-5 ページの「単一の単語または語句のローカライズ」を参照してください。
   - 長いテキスト ブロックをローカライズするには、3-5 ページの「長いテキスト ブロックのローカライズ」を参照してください。
3. `WEB-INF/catalogs` ディレクトリ内の `index.xml` ファイルにこのカタログのエントリを作成します。この手順については、3-6 ページの「index.xml ファイルの記述」を参照してください。

単一の単語または語句のローカライズ

単一の単語または語句のローカライズ テキストを作成するには、次の手順に従います。

1. <textlist> 要素を使用します。<textlist> 要素の中には、ローカライズ テキストの複数のエントリを名前 / 値のペアとして次の形式で作成できます。

   textid = localized text string

2. 次の XML を使用して、ローカライズされたテキストのすべてのエントリを、<textlist> 要素で囲みます。

   ```
   <![CDATA[
   ...
   ]]>
   ```

次に例を示します。

```
<textlist>
<![CDATA[

example.mytext      = This is localized text from the catalog.
example.title       = Console Extensibility Example
example.tab.extra   = Extra
example.tab.1       = Extension Catalog Text
example.tab.2       = Console Catalog Text
example.tab.3       = Programmatic
]]>
</textlist>
```

長いテキスト ブロックのローカライズ

長い文字列のローカライズ テキストを作成するには、次の手順に従います。

1. <text> 要素の id 属性を使用して、ローカライゼーション カタログ ID を指定します。次に例を示します。

   ```
   <text id='example.error-message'>
   ```

2. <text> タグと </text> タグの間にローカライズ テキストを挿入します。このテキストの中では HTML タグを使用できます。

3. ローカライズ テキストを次の XML で囲みます。

```
<![CDATA[
...
]]>
```

次に例を示します。

```
<text id='example.error-message'>
<![CDATA[
An <b>error</b> has occured.
<p> Please consult the documentation for more information.
]]>
</text>
```

index.xml ファイルの記述

index.xml ファイルは、言語とカタログ ファイルにマップし、ローカライズ テキストの表示に関する詳細を提供します。

index.xml ファイルを作成するには、次の手順に従います。

1. index.xml ファイルをプレーン テキスト ファイルとして作成し、コンソール拡張が格納されている Web アプリケーションの WEB-INF/classes ディレクトリに保存します。

   カタログ ファイルには、以下の基本 XML 要素を記述する必要があります。

   ```
   <?xml version="1.0" ?>
   <index>

   (Insert your catalog definitions here.)

   </index>
   ```

2. 定義するカタログごとに <catalog> 要素を作成します。この要素の中には、以下の属性を定義します。

属性	説明
name	言語の名前。詳細については、3-2 ページの「コンソールで使用するローカライゼーション カタログの決定」を参照。
file	この言語用に使用するカタログ ファイルのパスとファイル名。index.xml ファイルを起点とする相対パスを指定する。index.xml ファイルは、コンソール拡張が格納されている Web アプリケーションの WEB-INF/catalogs ディレクトリに存在する必要がある。
charset	ローカライズ テキストを表示するための文字セットの名前。World Wide Web Consortium（W3C）によって定義された標準文字セット名を使用する。
encoding	ローカライズ テキストを表示するためのエンコーディングの名前。World Wide Web Consortium（W3C）によって定義されたエンコーディングを使用する。

3. <catalog> 要素の中に 1 つまたは複数の <locale> サブ要素を作成します。Administration Console アプリケーションは、ユーザの Web ブラウザの設定値とこれらの属性を比較して、どのカタログを使用するかを決定します。詳細については、3-2 ページの「コンソールで使用するローカライゼーション カタログの決定」を参照してください。

属性	説明
country	次の URL で指定されている国。 http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt
language	次の URL で指定されている言語。 http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

4. コンソール拡張の言語カタログごとに <catalog> 要素と <locale> 要素を作成します。

サンプル ローカライゼーション カタログ

コード リスト 3-1　english.xml

```
<?xml version="1.0" ?>


<catalog>

<text id='example.error-message'>
<![CDATA[
An <b>error</b> {0} has occured.
<p> Please consult the documentation for more information.
]]>
</text>


<textlist>
<![CDATA[

example.mytext      = This is localized text from the catalog.
example.title         = Console Extensibility Example
example.copyright     = Copyright 2001 BEA Systems.
example.dialog        = This is an example dialog for the server
named {0}.
example.tab.extra     = Extra
example.tab.1         = Extension Catalog Text
example.tab.2         = Console Catalog Text
example.tab.3         = Programmatic
example.tab.4         = Switch on the Fly
]]>
</textlist>
</catalog>
```

JSP でのローカライズ テキストの使い方

以下の節では、コンソール拡張のさまざまな要素をローカライズするための JSP コードについて説明します。

ナビゲーション ツリー ノードのローカライズ

ナビゲーション ツリーのノードをローカライズするには、<wl:node> タグの labelId 属性を使用します。詳細については、2-2 ページの「<wl:node> タグ」を参照してください。

たとえば、次のコードを記述すると、ローカライゼーション カタログから node1 という ID で検索されるラベルを持つノードが作成されます。

```
<wl:node
   labelId='node1'
   icon='/images/bullet.gif'
   url='/dialog_domain_example.jsp'>
</wl:node>
```

右クリック メニューのローカライズ

右クリック メニューをローカライズするには、<wl:menu> タグの labelId 属性を使用します。詳細については、2-4 ページの「<wl:menu> および <wl:menu-separator> タグ」を参照してください。

たとえば、次のコードを記述すると、右クリック メニュー付きのノードが作成されます。このメニュー アイテムのラベルは、ローカライゼーション カタログから beaDocs と beaHome という ID で検索されます。

```
<wl:node
   label='My 2st nested node'
   icon='/images/bullet.gif'>
   <wl:menu
      labelId='beaDocs'
      url='http://e-docs.bea.com/index.html'
      target='_blank'/>
```

```
   <wl:menu-separator/>

    <wl:menu
        labelId='beaHome'
        url='http://www.bea.com'
         target='_blank'/>

</wl:node>
```

タブ ラベルのローカライズ

タブ ダイアログのラベルをローカライズするには、<wl:tab> タグの labelId 属性を使用します。詳細については、2-6 ページの「<wl:tab> タグ」を参照してください。

たとえば、次のコードではタブのローカライズ ラベルが使用されます。このラベルは、ローカライゼーション カタログから tab.1 という ID で検索されます。

```
<wl:tab name='LocalizedTextTab' labelId='tab.1'>
      The tab label for this tab comes from the catalog.
</wl:tab>
```

labelId 属性を指定しない場合、コンソールはローカライゼーション カタログから次の形式のエントリを検索します。

```
tab + name
```

コンソール ダイアログのテキストのローカライズ

コンソール画面のテキストをローカライズするには、以下の 2 つの節で説明するように、<wl:text> タグの textId および textParamId 属性を使用します。このタグの詳細については、2-9 ページの「<wl:text> タグ」を参照してください。

テキストのローカライズ

テキストをローカライズするには、<wl:text> タグの textId 属性を使用してローカライゼーション カタログからテキストを検索します。たとえば、次のコードでは、ローカライゼーション カタログから ID LocalizedText.1 が検索されます。

```
<wl:tab name='LocalizedTextTab' labelId='tab.2'>
    <wl:text textId='LocalizedText.1'/>
</wl:tab>
```

パラメータのローカライズ

パラメータをローカライズすることもできます。パラメータは、文字列 {0} の代わりに使用されます（文字列がローカライゼーション カタログに格納されている場合）。

たとえば、次のコードでは、LocalizedText.3 という ID で格納されている文字列 {0} の代わりに、LocalizedParam.1 という ID でカタログに格納されているローカライズ テキストが表示されます。

```
<wl:text
  textId='LocalizedText.3'
  textParamId='LocalizedParam.1' />
```