BEA WebLogic Server

WebLogic htmlKona
プログラマーズ ガイド
（非推奨）

WebLogic Server 6.1
マニュアルの日付：2001 年 11 月 30 日

著作権

限定的権利条項

商標または登録商標

WebLogic htmlKona プログラマーズ ガイド

マニュアルの版数	日付	ソフトウェアのバージョン
	2001 年 11 月 30 日	BEA WebLogic Server バージョン 6.1

目次

このマニュアルの内容

対象読者 vii
e-docs Web サイト vii
このマニュアルの印刷方法 viii
関連情報 viii
サポート情報 viii
表記規則 ix

1. WebLogic htmlKona の概要

WebLogic htmlKona の非推奨 1-1
htmlKona とは 1-1
補足情報 1-2

2. htmlKona の使い方

非推奨になった htmlKona の使い方 2-1
htmlKona の概要 2-2
    htmlKona オブジェクトとそれらのクラス 2-3
    要素 : htmlKona のペイント ツール 2-4
htmlKona を使った HTML ドキュメントの生成 2-4
    ページの設定 2-5
        HtmlPage の作成 2-7
        HEAD 要素の付加 2-7
        BODY 要素への属性の付加 2-8
        ドキュメント本文の作成 2-9
        例 2-9
    その他のページ定義要素 2-10
    ブロック属性の設定 2-10
    物理属性と論理属性の設定 2-11
    その他のテキストレベル クラス 2-13

AddressElement 2-13
SpacerElement 2-13
カプセル化クラス 2-14
Codeset 2-14
AlignType 2-15
TexAlignType 2-15
ClearType 2-16
HtmlColor 2-16
WindowName 2-16
アンカーの作成 2-16
リストの使い方 2-18
DefinitionList 2-18
ListElement 2-19
画像の使い方 2-20
サーバサイドおよびクライアントサイドの画像マップの使い方 2-20
フレームの使い方 2-23
FrameElement 2-23
FrameSetElement 2-23
NoFramesElement 2-24
ScrollType 2-24
例 2-24
テーブルの使い方 2-25
例 2-26
フォームの設定 2-27
フォーム データの取得 2-29
例 2-29
ページへのスクリプトの付加 2-31
ScriptElement の使い方 2-31
例 2-31
ページへのアプレットの付加 2-33
AppletElement 2-33
例 2-34
ページへのファイルの埋め込み 2-36
htmlKona を使った動的データの表示 2-36
Java 対応サーバでの htmlKona の使い方 2-41

Java サーバサイド サーブレットの例.....2-41
出力テストの簡単な方法.....2-44

このマニュアルの内容

このマニュアルでは、BEA WebLogic Server™ htmlKona について説明します。htmlKona は、HTML 環境とのオブジェクト指向インタフェースを提供します。このインタフェースを使用すれば、オブジェクトを使って HTML ドキュメントのフォーマットを設定できるようになります。この製品は非推奨になり、現在はサポートされていません。

このマニュアルの内容は以下のとおりです。

- 第 1 章「WebLogic htmlKona の概要」では、JavaServer Pages（JSP）の WebLogic 実装とそのアーキテクチャについて概説します。
- 第 2 章「htmlKona の使い方」では、WebLogic htmlKona の機能について説明します。

対象読者

このマニュアルは、複雑な HTML ドキュメントをプログラムで生成するプロセスの設定が必要な方を対象としています。Web 技術、オブジェクト指向プログラミング技術、Java プログラミング言語について読者が精通し、HTML およびページを表示するブラウザについての実践的知識があることを前提としています。

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 ホームページの [製品のドキュメント] をクリックするか、WebLogic Server 製品ドキュメント ページ（http://edocs.beasys.co.jp/e-docs/wls61）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 章ずつ印刷できます。

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 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

BEA WebLogic の htmlKona 製品は非推奨になりました。現在、WebLogic では、WebLogic JavaServer Pages（JSP）による JSP のサポートを提供しています。Java 2 Enterprise Edition 標準に従って、JSP は Web アプリケーションの一部としてデプロイされます。Web アプリケーションとは、HTTP サーブレット、JavaServer Pages（JSP）、静的 HTML ページ、画像、その他のリソースなどからなる一群のアプリケーション コンポーネントのことです。JSP の詳細については、『WebLogic JSP プログラマーズ ガイド』を参照してください。

サポート情報

BEA 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@bea.com までお送りください。寄せられた意見については、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、ご使用のソフトウェアの名前とバージョン、およびドキュメントのタイトルと日付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
〔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。

表記法	適用

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:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

斜体の等幅テキスト	コード内の変数を示す。 例: `String CustomerName;`
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 BEA_HOME OR
{ }	構文の中で複数の選択肢を示す。
[ ]	構文の中で任意指定の項目を示す。 例: `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`
|	構文の中で相互に排他的な選択肢を区切る。 例: `java weblogic.deploy [list|deploy|undeploy|update] password {application} {source}`

表記法	適用
. . .	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
. . .	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。

1 WebLogic htmlKona の概要

WebLogic htmlKona の非推奨

WebLogic htmlKona は、WebLogic Server 6.x では非推奨になっています。

以下の節では、WebLogic htmlKona の概要と、WebLogic Server での JSP の機能について説明します。

- htmlKona とは
- 補足情報

htmlKona とは

htmlKona クラスを使うと、複雑な HTML ドキュメントをプログラムで簡単に作成できるようになります。htmlKona は、他のスクリプト言語や多くのブラウザ拡張機能をサポートしているので、対話的な HTTP サーブレット環境でも、静的 HTML ページを定期的に作成する場合にも役に立ちます。

htmlKona は、クエリから動的ページを生成できるので、他のデータベース関連およびイベント関連の WebLogic 製品を補う強力なツールとなります。多層環境では、htmlKona は異なる種類のデータベースからオーディオ クリップや GIF/JPEG 画像などのマルチメディア オブジェクトを取得して、HTML ページに組み込むことができます。htmlKona は、WebLogic などの Java 対応 HTTP サーバに対して実行されたときにクライアント ブラウザ向けの HTML ページを生成するクラス ファイルを作成します。

補足情報

- 『WebLogic HTTP サーブレット プログラマーズ ガイド』
- 「Web アプリケーションのデプロイメント記述子の記述」

2 htmlKona の使い方

以下の節では、WebLogic htmlKona の概要およびその機能について説明します。

- 非推奨になった htmlKona の使い方
- htmlKona の概要
- htmlKona を使った HTML ドキュメントの生成

非推奨になった htmlKona の使い方

このマニュアルは更新されていませんが、このリリースと WebLogic Server バージョン 5.1 の主な相違点を次の表に示します。

表 2-1 非推奨になった htmlKona のリソース

使用する機能	従来の機能	説明
myDriver.connect()	DriverManager.getConnection()	`DriverManager.getConnection()` は同期されるメソッドで、ある状況でアプリケーションにハングを引き起こします。このため、`DriverManager.getConnection()` の代わりに **`Driver.connect()`** メソッドを使用することをお勧めします。
Administration Console	weblogic.properties ファイル	Administration Console を使用して属性を設定します。このツールは weblogic.properties ファイルに取って代わるものです。

htmlKona の概要

htmlKona クラスを使うと、複雑な HTML ドキュメントをプログラムで簡単に作成できるようになります。htmlKona は、他のスクリプト言語や多くのブラウザ拡張機能をサポートしているので、対話的な HTTP サーブレット環境でも、静的 HTML ページを定期的に作成する場合にも役に立ちます。

htmlKona は、クエリから動的ページを生成できるので、他のデータベース関連およびイベント関連の WebLogic 製品を補う強力なツールとなります。多層環境では、htmlKona は異なる種類のデータベースからオーディオ クリップや GIF/JPEG 画像などのマルチメディア オブジェクトを取得して、HTML ページに組み込むことができます。htmlKona は、WebLogic などの Java 対応 HTTP サーバに対して実行されたときにクライアント ブラウザ向けの HTML ページを生成するクラス ファイルを作成します。

htmlKona は、HTML 環境とのオブジェクト指向インタフェースを提供します。このインタフェースを使用すれば、オブジェクトを使って HTML ドキュメントのフォーマットを設定できるようになります。htmlKona には、2 種類のオブジェクトが存在します。すなわち、ページ（weblogic.html.WebPage から派生）と、それに付加される要素（weblogic.html.HtmlElement から派生）です。htmlKona では、WebPage はキャンバスのように扱われます。タグ付き構文を使ってマークアップする代わりに、ページ オブジェクトを作成し、それに htmlKona 要素を追加するのです。それが完了したら、そのページ オブジェクトの `output()` メソッドの 1 つを呼び出すことで、ページが表現されます。

htmlKona 要素の特徴

- htmlKona 要素の中には、テキスト マークアップ用の物理要素や論理要素のように、シングルパートのものがあります。たとえば、BoldElement、ItalicElement、StrongElement などです。それ以外の要素（UnorderedList など）はマルチパートです。最初に中身のないマルチ パート要素を作成してから、特定のシングルパート要素を使って組み立てます。たとえば、UnorderedList は ListItems を付加することによって作成します。
- htmlKona 要素の中には、属性を備えているものがあります（たとえば、FontElement には色、サイズ、フォント名の各属性を設定できます）。属性そのものはカプセル化されることがあります。たとえば、16 種類の HTML 色

の名称は HtmlColor にカプセル化され、オブジェクトを整列させるさまざまな方法は AlignType にカプセル化されます。

- htmlKona 要素の中には、コンポーネントを持っているものがあります（HtmlContainer など）。

このマニュアルは、HTML についても、Web ページを表示するブラウザについても十分な実践的知識を持っているユーザを対象としています。HTML のチュートリアルや入門書ではありません。htmlKona は多くのブラウザ拡張機能をサポートしていますが、ここでは、自分のブラウザがどの拡張機能をサポートしているかを知っているものと想定しています。

htmlKona オブジェクトとそれらのクラス

htmlKona では要素が階層的に整理されているので、HTML の作成にオブジェクト指向環境の構造と能力を利用することができます。htmlKona には、WebPage と HtmlElement という 2 つの主要な派生系統があります。

どの htmlKona ページでも、まず htmlKona の WebPage のサブクラスを作成します。ほとんどの場合、HTTP サーブレット内で htmlKona を使って **ServletPage** を作成します。htmlKona ページを作成したら、htmlKona の HtmlElement をそのページのヘッダ コンテナ（`getHead()` で取得）か、本文コンテナ（`getBody()` で取得）に追加します。これらのコンテナは、それぞれ wblogic.html.HtmlContainer です。これらの要素は順序付けられており、1 つまたは複数の HtmlContainer の内部で指定されることがあります。HtmlContainer は、それ自体 HtmlElement です。

ページ : htmlKona のキャンバス

すべての htmlKona ページは、weblogic.html.WebPage クラスから派生します。以下の 4 種類の htmlKona WebPage が存在します。

- オーディオ ページ
- 画像ページ
- プレーン テキスト ページ
- HTML ページ。**ServletPage** はその特殊なケース

htmlKona は、javax.servlet.http.HttpServlet のサブクラスである HTTP サーブレットと一緒に使われる場合がほとんどです。

要素 : htmlKona のペイント ツール

すべての htmlKona 要素は、weblogic.html.HtmlElement クラスから派生します。以下の 5 種類の一般的な htmlKona Element が存在します。

1. ElementWithAttributes
2. FileElement（めったに使用されない）
3. HtmlContainer（それ自身が HtmlElement）
4. MarkupElement（さまざまなマークアップ要素をカプセル化したもの）
5. StringElement（HtmlElement の特殊なケース）

weblogic.html.ElementWithAttributes は、さらに weblogic.html.MultiPartElement と weblogic.html.SinglePartElement という 2 つのサブクラスに分類されます。ElementWithAttributes には、WebPage を作成するために使用する要素の大部分が含まれています。

htmlKona を使った HTML ドキュメントの生成

htmlKona に含まれるクラスは、機能別あるいは派生関係の階層別に編成することができます。この節では、クラスを機能別にまとめて説明します。たとえば、リストを作成するのに必要と思われる要素は、すべて同じトピックで説明します。

- ページの設定
  - HtmlPage の作成
  - HEAD 要素の付加
  - BODY 要素への属性の付加

- ドキュメント本文の作成
- 例

- その他のページ定義要素
- ブロック属性の設定
- 物理属性と論理属性の設定
- その他のテキストレベル クラス
- カプセル化クラス
- アンカーの作成
- リストの使い方
- 画像の使い方
- サーバサイドおよびクライアントサイドの画像マップの使い方
- フレームの使い方
- テーブルの使い方
- フォームの設定
- ページへのスクリプトの付加
- ページへのアプレットの付加
- ページへのファイルの埋め込み
- htmlKona を使った動的データの表示
- Java 対応サーバでの htmlKona の使い方
  - Java サーバサイド サーブレットの例
- 出力テストの簡単な方法

ページの設定

htmlKona を使って Web ページを作成するには、まず WebPage オブジェクトを構築し、必要に応じてその HEAD 要素とその他の属性を設定します。次に、WebPage に HtmlElement を追加して、ドキュメントを作成します。htmlKona

ファイルをコンパイルしたら、それを Java 対応サーバ上の適切なディレクトリに置き、サーバを再起動（場合によって）し、ブラウザから URL を指定して、サーバにそのページを要求します。

WebPage には、以下の 5 つのタイプがあります。これらはすべて、親クラス weblogic.html.WebPage から派生するサブクラスです。

- weblogic.html.HtmlPage: HTML 要素を追加することによって作成される通常の HTML タグ付きドキュメントを作成します。Content-type ヘッダをページに追加し、希望する場合は、スーパクラスからもたらされるメソッドを使ってその他のヘッダ情報を設定できるようになります。
- weblogic.html.ServletPage: サーバサイド Java に適したページ、すなわち Content-type ヘッダのないページを作成します。サーバサイド Java では、使用しているサーバによって決まる他のメソッドでヘッダ情報が与えられます。
- weblogic.html.AudioPage: audio タイプと、ページ作成用のオーディオ データであるバイト配列で構成されます。
- weblogic.html.ImagePage: image タイプと、ページ作成用の画像データであるバイト配列で構成されます。
- weblogic.html.PlainPage: 非 HTML タグ付きページを作成します。HTML フォーマッティングは行われず、テキストがそのままの状態で表示されます。

HtmlPage は、content-type が「text/html」であり、最も一般的に使われるものです。AudioPage と ImagePage は、データベースから取得された BLOB オーディオやビジュアル画像を含んだマルチメディア HTML ドキュメントを作成するために使用されます。PlainPages は、書式付けされていないテキストを表示するために使用されます。ServletPage は、HtmlPage のサブクラスですが、content-type は設定されません。これらのページの違いは、HTTP サーバに渡される content-type だけです。

WebPage スーパクラスには、コンテンツのエンコーディング、データ長、およびタイプを設定するメソッド、位置、サーバ、参照元、およびプラグマを設定するメソッド、有効期限と最終変更日時を設定するメソッド、そしてその他の属性を設定するメソッドがあります。

WebPage の各タイプには、取り扱う HtmlElement のタイプを処理するメソッドがあります。

HtmlPage の作成

最も一般的に使われる WebPage オブジェクトは、ServletPage です。これは HtmlPage のサブクラスで、Java-Servlet-API スタイルの HTTP サーブレットと一緒に使用されます。HtmlPage は、HtmlElement を使ってページの内容をレイアウトするための htmlKona キャンバスです。レイアウトが完了したら、output() メソッドを呼び出します。すると、そのページの HTML が自動的に生成されます。

最初に、ServletPage を作成します。

```
ServletPage hp = new ServletPage();
```

また、String 型の引数としてタイトルを指定してページを作成することもできます。

ページのコード セットを設定するには、Codeset を引数として取る ServletPage コンストラクタを使います。次に例を示します。

```
ServletPage sp = new ServletPage(new CodeSet("SJIS"));
```

HEAD 要素の付加

htmlKona は HTML の HEAD 要素をすべてサポートしており、これらの要素はサーバとブラウザの双方に情報を提供します。これらをページに追加するには、`HtmlPage.getHead()` メソッドを呼び出してから、`addElement()` メソッドを使います。

```
hp.getHead()
   .addElement(new TitleElement("htmlKona test2"))
   .addElement(new MetaElement(MetaElement.nameType,
                               MetaElement.nameDescription,
                               "WebLogic test meta element"))
   .addElement(new MetaElement(MetaElement.equivType,
                               MetaElement.httpEquivExpires,
                               new java.util.Date()))
   .addElement(new MetaElement(MetaElement.nameType,
                               MetaElement.nameRobots,
                               MetaElement.indexNoFollow))
   .addElement(new LinkHeadElement(LinkHeadElement.relTag,
                               LinkHeadElement.relHome,
                               "www.weblogic.com",
"WebLogic Home"));
```

HEAD 要素用の htmlKona クラスは以下のとおりです。

- weblogic.html.TitleElement（HTML ページ上で必要な唯一の要素）
- weblogic.html.IsIndexElement
- weblogic.html.LinkHeadElement
- weblogic.html.MetaElement
- weblogic.html.ScriptElement
- weblogic.html.StyleElement
- weblogic.html.BaseElement
- weblogic.html.BaseFontElement（ブラウザ拡張機能）

MetaElement と LinkHeadElement の両方には、これらの要素が取り得る名前と値をカプセル化した定数が含まれています。また、ページ ヘッダに対するブラウザ拡張機能である BaseFontElement もサポートされています。これを使用すると、ドキュメント全体のデフォルトのフォント サイズを設定できるようになります。

BODY 要素への属性の付加

ページの HEAD 部分を設定したら、`HtmlPage.getBodyElement()` メソッドを呼び出して、BODY 要素の属性を設定します。属性を設定するには、BodyElement クラスから提供される `setAttribute()` メソッドを使います。このクラスは、現在の BODY 属性用の一連の定数も提供します。

```
hp.getBodyElement()
  .setAttribute(BodyElement.bgColor, "#FFFFFF")
  .setAttribute(BodyElement.backgroundImg,
                "images/wtbkg.gif")
  .setAttribute(BodyElement.linkColor, HtmlColor.fuchsia)
  .setAttribute(BodyElement.aLinkColor, "#080808")
  .setAttribute(BodyElement.vLinkColor, "#808080");
```

また、スクリプト用の ONLOAD 属性と ONUNLOAD 属性もサポートされています。16 個の HTML カラー名が weblogic.html.HtmlColor クラスにカプセル化されていることに注意してください。色に関連する htmlKona 操作はすべて、引数として java.awt.Color オブジェクトを取るメソッドも提供します。

ドキュメント本文の作成

HtmlPage を作成し、その HEAD 要素を追加し、そしてそのページの BODY 属性を設定したら、さまざまな `addElement()` メソッドを使ってページに HtmlElement を追加できます。それが完了したら、`HtmlPage.output()` メソッドのうちの 1 つを使って、適切なソースにページを出力します。

例

ここで、htmlKona を使って作成したクラスの完全なコードの非常に簡単な例を示します。このクラスは `main` を付けて書かれているので、Java 対応サーバに対してこのクラスを実行せずに HTML を出力できるようになります。

```
import java.io.*;
import weblogic.html.*;

public class helloworld {

  public static void main(String argv[])
    throws HtmlException, IOException
  {
    HtmlPage hp = new HtmlPage();
    hp.getHead()
      .addElement(new TitleElement("The Hello World Page"));
    hp.getBodyElement()
      .setAttribute(BodyElement.bgColor, HtmlColor.white)
      .setAttribute(BodyElement.backgroundImg,
                    "images/mylogo.gif");
    hp.getBody()
      .addElement(MarkupElement.HorizontalRule)
      .addElement(new StringElement("Hello World!")
                  .asBoldElement())
      .addElement(MarkupElement.HorizontalRule);
    hp.output();
  }
}
```

このサンプルの HTML は、以下のようになります。

Hello World!

この例をテストするには、次の手順を実行します。

1. サンプルのコードを `helloworld.java` というファイルにコピーします。
2. それをコンパイルします。
3. クラス ファイルを `CLASSPATH` に置きます。

4. `java helloworld > test.html` を使って HTML 出力を作成します。
5. その HTML ファイルをブラウザで表示します。

その他のページ定義要素

htmlKona では、フレームを作成することもできます。フレームは、すべてのブラウザで表示できるわけではありません。htmlKona は、フレームを幅広くサポートしています。

ブロック属性の設定

HTML 仕様では、マークアップ要素は 2 つの一般クラスに分かれます。すなわち、パラグラフ区切りを生じさせるもの（ブロックレベル）と、生じさせないもの（テキストレベルの物理要素と論理要素の両方）です。htmlKona ではさらに、複数の部分から成るブロックレベル要素（リストやフォームなど）と、テキストやその他のテキストレベルの要素だけを含むものを明確に区別しています。このマニュアルでの説明は、HTML そのものが使用している構成にできるだけ準拠するように記述してあるので、HTML に精通していれば、htmlKona が理解しやすいことがわかるでしょう。したがって、最初に単純なブロックレベル要素について説明し、次に物理要素と論理要素、さらにテーブルやフォームといったより複雑な要素へと進むことにします。

htmlKona に含まれる単純なシングルパートのブロックレベル要素には、以下のものがあります。

- weblogic.html.BlockquoteElement
- weblogic.html.BreakElement
- weblogic.html.CenteredElement
- weblogic.html.DivElement
- weblogic.html.HeadingElement
- weblogic.html.HorizontalRuleElement
- weblogic.html.LiteralElement
- weblogic.html.ParagraphElement

各クラスのメソッドを調べて、これらのオブジェクトにさらに設定できる属性を確かめてください。これらのオブジェクトの使い方は非常に簡単です。HtmlPage の本文部分と同じように、新しいオブジェクトを作り、それを HtmlContainer に追加すればよいのです。たとえば、「hp」という HtmlPage の本文部分に見出し（H2）と水平の罫線を追加する方法は、以下のとおりです。

```
hp.getBody()
  .addElement(new HeadingElement("Breaking New Ground", 2))
  .addElement(new HorizontalRuleElement());
```

これらの要素の多く、特に終了タグを必要としないものは、一般 htmlKona クラスである weblogic.html.MarkupElement の中にカプセル化されています。たとえば、HtmlPage に水平の罫線を挿入するには、以下の 2 つの方法があります。

```
hp.getBody()
  .addElement(new HorizontalRuleElement()
              .setAlign(AlignType.center)
              .setWidth("50%"))
```

または、単に以下のようにします。

```
hp.addElement(MarkupElement.HorizontalRule)
```

最初の方法では、HorizontalRule オブジェクトを作成するので、HorizontalRule クラスの他のメソッドを使って、オブジェクトにさらに属性を設定することができます。さらに属性を設定するつもりがない場合は、2 番目の方法を使うと、ページに簡単にマークアップ要素を追加できます。

物理属性と論理属性の設定

物理属性と論理属性用のクラス

- weblogic.html.BigElement
- weblogic.html.BoldElement
- weblogic.html.CiteElement
- weblogic.html.CodeElement
- weblogic.html.CommentElement
- weblogic.html.DefineTermElement
- weblogic.html.EmphasisElement

- weblogic.html.FontElement
- weblogic.html.ItalicElement
- weblogic.html.KeyboardElement
- weblogic.html.SampleElement
- weblogic.html.SmallElement
- weblogic.html.StrikeElement
- weblogic.html.StrongElement
- weblogic.html.SubscriptElement
- weblogic.html.SuperscriptElement
- weblogic.html.TeletypeElement
- weblogic.html.UnderlineElement
- weblogic.html.VariableElement

物理属性と論理属性は、汎用的な方法で扱われます。新しいオブジェクトを作成し、それを HtmlPage（または任意の HtmlContainer）に追加し、そのオブジェクトの `setXXX()` メソッドを使ってさらに属性を設定します。場合によっては、別のクラスのメソッドを使ってさらに属性を設定するために、HtmlElement を別のタイプの要素としてキャストすることができます。

この例では、FontElement を作成し、そのサイズと色を設定しています。

```
hp.getBody()
    .addElement(new FontElement(5, HtmlColor.red,
                                "To the Stars!"));
```

また、別の HtmlElement を FontElement としてキャストし、その HtmlElement に対して FontElement クラスのメソッドを使うこともできます。たとえば、以下の例では、要素を中央に配置して、その色を設定しています。

```
  hp.getBody()
    .addElement(new HeadingElement(filename, 2)
                .setAlign(TextAlignType.center)
.asFontElement("3", HtmlColor.fuchsia));
```

その他のテキストレベル クラス

その他に、パラグラフ区切りを生じさせず、特に物理要素あるいは論理要素というわけでもないテキストレベル クラスが、以下のように 2 つあります。

- weblogic.html.AddressElement
- weblogic.html.SpacerElement

AddressElement

AddressElement クラスは、ページの作成者を明らかにするもので、また、連絡先情報用のハイパーリンクを作成するためのコンストラクタも備えています。多くの場合、これはイタリック体のテキストで表示されます。次に例を示します。

```
hp.getBody()
  .addElement(new AddressElement("http://www.weblogic.com",
                "BEA Systems Inc."));
```

AddressElement を使って複数行にわたる住所をページに追加する場合には、各行の間に BreakElement（または MarkupElement.Break）を追加しなければなりません。次に例を示します。

```
HtmlContainer ae = new HtmlContainer();
ae.addElement("BEA Systems Inc.")
  .addElement(MarkupElement.Break)
  .addElement("180 Montgomery Street, Suite 1240")
  .addElement(MarkupElement.Break)
  .addElement("San Francisco, California 94104")
  .addElement(MarkupElement.Break);
hp.getBody()
  .addElement(new AddressElement(ae));
```

SpacerElement

SpacerElement（一部のブラウザでサポートされている）を使うと、ページ上のオブジェクトの間に水平空白、垂直空白、あるいはブロック空白を設定できるようになります。サイズ属性か、あるいは幅、高さ、および位置合わせ属性のいずれかが設定されます。SpacerElement のコンストラクタでは、weblogic.html.SpacerType オブジェクトを使って、SpacerElement の type 属性を設定してもかまいません。

カプセル化クラス

htmlKona では、他のオブジェクトで使われることの多いある種の属性をカプセル化するためのクラスをいくつか提供しています。Codeset クラスと HtmlColor インタフェースを除き、カプセル化クラスはすべて MarkupElement のサブクラスです。ここでは、AlignType や HtmlColor のような一般に使われるクラスについて説明します。特定の要素あるいは要素群に関連づけられている他のクラスについては、このマニュアル内の別の箇所（リンクされる）で、その要素と一緒に説明します。以下のカプセル化クラスがあります。

- CodeSet（ServletPage と HtmlPage のコンストラクタで使用される）
- AlignType
- AnchorType（AnchorElement で使用される）
- BorderstyleType（一部のブラウザにおいて TableElement で使用される）
- ClearType
- FrameType（一部のブラウザにおいて TableElement で使用される）
- FieldType（InputElement で使用される）
- HtmlColor
- RulesType（一部のブラウザにおいて TableElement で使用される）
- ScrollType（FrameElements で使用される）
- SpacerType（SpacerElement で使用される）
- TextAlignType
- WindowName
- WrapType（TextAreaElement で使用される）

Codeset

weblogic.html.Codeset

オブジェクトを ServletPage または HtmlPage の引数として使用して、ドキュメントのコード セットを設定します。設定しない場合のデフォルトのコード セットは「8859_1」です。これは、英語、ドイツ語、ロマンス諸語で使われるラテン文字セットです。次に例を示します。

```
ServletPage sp = new ServletPage(new Codeset("SJIS"));
```

この例では、コード セットを日本語の Shift-JIS に設定しています。コード セットは、JavaSoft の JavaSoft 国際化仕様書で定義されています。

AlignType

weblogic.html.AlignType

オブジェクトをさまざまな setAlign() メソッドの引数として使用して、AppletElement、HorizontalRuleElement、およびさまざまな Table*Element などの HtmlElement の位置合わせを設定します。このクラスに定義されている変数が、位置合わせを設定します。

注意： コンテナ内のテキストの位置合わせには、AlignType ではなく TextAlignType が使用されることに注意してください。

次に、AlignType オブジェクトを使ってテーブルの各部分の位置合わせを設定する例を示します。

```
TableElement table = new TableElement();
table.setCaption(new TableCaptionElement("Usage Statistics")
                    .setAlign(AlignType.bottom))
     .setBorder(1)
     .addElement(new TableRowElement()
                 .addElement(new TableDataElement(reg.getID())
              .setVAlign(AlignType.top))
                  .addElement(new TableDataElement(
                new TableElement(reg.getMain())
                                      .setCaption(
         new BoldElement("N1"))
                                      .setBorder(1))
                                  .setVAlign(AlignType.top)
     .setAlign(AlignType.right)));
```

TexAlignType

weblogic.html.tTextAlignType

オブジェクトは、ParagraphElement、DivElement、HeadingElement といった HtmlElement のテキスト コンテンツの位置合わせを設定するために使用されます。このクラスの変数には、center（中央揃え）、justify（行揃え）、left（左揃え）、および right（右揃え）があります。TextAlignType オブジェクトは、テキスト コンテナの setAlign() メソッドの引数として使います。

ClearType

weblogic.html.ClearType

オブジェクトは HTML の CLEAR 属性を設定します。これによって、HtmlElement（BreakElement または DivElement）は、図やテーブルの横ではなく、それらの後に無条件に移動します。

HtmlColor

16 個の HTML カラー名は、String 要素としてクラス weblogic.html.HtmlColor 内にカプセル化されるので、色を引数に取るメソッドやコンストラクタは、HtmlColor クラス内の変数（HtmlColor.white など）を使うか、あるいは 6 文字の文字列から成る 16 進の RGB 値（「#FFFFFF」など）を使用できます。

注意： 通常、引数として java.awt.Color オブジェクトも取る同一のメソッドやコンストラクタが存在します。

WindowName

weblogic.html.WindowName オブジェクトを使って、さまざまな htmlKona クラスにおける setTarget() メソッドでリンクのターゲット ウィンドウを設定します。このクラス内の変数は、HTML のマジック ターゲット名である _blank、_parent、_self、および _top に似ています。リンクのターゲットは、多くの場合、フレーム、画像、画像マップ、およびアンカー要素で使用されます。

アンカーの作成

アンカー作成用のクラス

- AnchorElement

■ AnchorType

weblogic.html.AnchorElement オブジェクトを使って、HtmlPage にハイパーテキスト リンクを設定します（これは、旧クラス LinkElement に取って代わるものですが、下位互換性を保つため、LinkElement も依然としてサポートされています）。アンカーの実際の用途は 2 つあります。ひとつは HREF 属性で使って、別のドキュメントまたは内部リンクへのハイパーリンクを設定すること、もうひとつは、NAME 属性で使って、内部リンクを識別することです。たとえば、このパラグラフには <ANAME="anchorelementdef"> というアンカーが付いています。これにより、このドキュメント内で、<A HREF=#anchorelementdef> というアンカーが付いているリンクから、このパラグラフにジャンプできるようになるのです。

したがって、このクラスのコンストラクタには 2 つのタイプがあります。1 つは主に HREF 属性でテキストをハイパーリンク化するためのもので、もう 1 つは主に NAME 属性でテキストをマーキングするためのものです。また、このクラスのコンストラクタは、カプセル化クラス weblogic.html.AnchorType のオブジェクトをコンストラクタの引数として使用して、HREF 属性と NAME 属性に単なる String を使う場合に起こるあいまいさを解消します。実際、同じアンカー内で HREF 属性と NAME 属性の両方を使用して、同一ドキュメント内でのリンク先を提供すると同時に、別のドキュメントにリンクすることができます。このパラグラフのソースを見れば、上記の AnchorType クラス名のハイパーリンクで両方の属性が使用されていることがわかります。

また、AnchorElement コンストラクタのひとつで weblogic.html.WindowName オブジェクトを使って、指定された URL を別のブラウザ ウィンドウに送ることもできます。

ここで、AnchorElement の使い方をいくつか示します。

```
hp.getBody()
  .addElement(
    new AnchorElement("http://www.acme.com/copyright.html",
                      "© 1996, Acme Inc."))
  .addElement(
    new AnchorElement(AnchorType.href,
                      "http://www.acme.com/copyright.html",
                      "© 1996, Acme Inc."))
  .addElement(
    new AnchorElement(AnchorType.name,
                      "topic1",
                      "The most important topic"))
  .addElement(
    new AnchorElement("http://www.acme.com/copyright.html",
                      "© 1996, Acme Inc.",
      WindowName.parent));
```

また、以下の例のように、HtmlPage.asAnchorElement() メソッドを使って、内部の NAME アンカーを設定することもできます。

```
hp.getBody()
  .addElement(new StringElement("This text is an " +
                                "internal anchor")
              .asAnchorElement("anchor1"))
  .addElement(new StringElement(" in this document."));
```

リストの使い方

リストには、DefinitionList と ListElement の 2 種類があります。htmlKona は、OrderedList と UnorderedList、MenuList、DirList など、ListElement のいくつかのサブクラスをサポートしています。HTML 3.2 では MenuList と DirList が非推奨になり、より一般的な UnorderedList に置き換えられたことに注意してください。htmlKona では、下位互換性を保つためにこれらをサポートしていますが、すべてのブラウザがこうしたリストを引き続きサポートするとはかぎりません。

リスト用のクラス

- weblogic.html.DefinitionList には、1 つまたは複数の weblogic.html.DefinitionItem が含まれています。
- weblogic.html.ListElement には、1 つまたは複数の weblogic.html.ListItem が含まれており、以下の種類があります。

weblogic.html.DirList

- weblogic.html.MenuList
- weblogic.html.OrderedList
- weblogic.html.UnorderedList

DefinitionList

DefinitionList は DefinitionItem から構成され、addElement() メソッドによって追加されます。DefinitionItem には、用語と定義という 2 つの部分があります。List 内の DefinitionItem は、インデックス位置によってアクセス可能です。

ListElement

ListElement にはいくつかの種類があり、それらは HTML の番号付きリストと番号なしリストに相当します。ListElement は、ListItem または文字列から成ります。リスト内の項目には、インデックス位置によってアクセス可能です。htmlKona は、リスト要素の追加属性をいくつかサポートしています。

次に、番号なしリストの作成例を示します。

```
UnorderedList ul = new UnorderedList();
ul.setType(UnorderedList.circle)
  .addElement(new ListItem("$500"))
  .addElement(new ListItem("$250"))
  .addElement(new ListItem("$100"));
```

この結果、以下のように出力されます。

- $500
- $250
- $100

htmlKona は、番号付きリストと番号なしリストの両方の type 属性をサポートしています。各種類のリストの有効タイプは、各クラスに String 型定数としてカプセル化されています。

また、htmlKona は、OrderedList および OrderedList に追加された ListItem のための start 引数もサポートしています。このため、以下の例のように、番号のシーケンスを制御できるようになります。

```
OrderedList ol = new OrderedList();
ol.setType(OrderedList.largeRoman)
  .setStart(13)
  .addElement("500")
  .addElement("600")
  .addElement("435")
  .addElement(new ListItem("250")
              .setType(OrderedList.smallRoman)
              .setValue("7"))
  .addElement("195")
  .addElement("100");
```

この結果、次のように表示されます。

1. 500

2. 600
3. 435
4. 250
5. 195
6. 100

画像の使い方

画像管理用のクラス

- weblogic.html.ImageElement

ImageElement を使って、HtmlPage 上にインライン画像を配置します。次に例を示します。

```
hp.getBody()
  .addElement(new ImageElement("http://website/images/bar.gif"));
```

ImageElement クラスには、ImageElement の高さ、幅、左右のマージン幅、上下のマージン幅、ソース、枠、代替テキスト、および位置合わせを設定するためのメソッドがあります。さらに、htmlKona はサーバサイドおよびクライアントサイドの両方の画像マップをサポートしています。そのために、ImageElement にはさらに、setUseMap() メソッドと setIsMap(boolean) メソッドがあります。

また、weblogic.html.ImagePage を使って、データベースからバイト配列（byte[]）として取得した画像を表示することもできます。

サーバサイドおよびクライアントサイドの画像マップの使い方

サーバサイド画像マップ用のクラス

- weblogic.html.AnchorElement

- weblogic.html.ImageElement

サーバサイド画像マップとは、サーバ上で CGI スクリプトを実行させて、画像領域に対するマウス イベントに応答するものです。ImageElement が画像マップとして動作するには、追加属性の ISMAP が必要です。サーバサイド画像マップを作成するには、CGI スクリプトを URL として設定する AnchorElement の引数として ImageElement を使用する必要があります。次に例を示します。

```
// ImageElement を作成する
ImageElement ie =
  new ImageElement("http://www.weblogic.com/images/h.gif");
// ISMAP 属性を設定する
ie.setIsMap(true);
hp.getBody()
  //  CGI スクリプトを URL とし ImageElement を表示する
  // AnchorElement を作成する
  .addElement(new AnchorElement("/cgi-bin/imagemap", ie));
```

現在、いくつかのブラウザでは、クライアントサイド画像マップもサポートしており、これを使えば、サーバサイドの情報（サーバサイド画像マップ）ではなく画像に関連付けられたローカルの情報を利用できるようになります。htmlKona はクライアントサイド画像マップをサポートしていますが、表示に使うブラウザがそれをサポートしているかどうかを確認する必要があります。

クライアントサイド画像マップ用のクラス

- weblogic.html.AreaElement
- weblogic.html.ImageElement
- weblogic.html.MapElement

MapElement は、その名前を設定することによって作成されるもので、AreaElement を付加することで作成されるマルチパートの要素です。AreaElement を使って、画像マップ上の各「ホット ゾーン」の座標を設定します。

AreaElement は、特定の URL にマップされる画像の領域の座標を設定するために使用されます。コンストラクタとメソッドでは、AreaElement クラス内に（String 型定数として）カプセル化されている有効な領域形状、または java.awt.Rectangle と java.awt.Polygon を使用できます。このクラスには、ほかにも座標と URL を設定するためのメソッドが含まれています。なお、画像が存在しないときや表示されない場合にユーザに何らかの情報を与えるため、ALT 属性を設定することをお勧めします。

以下の例は、スクリーン画像からヘルプへのリンクを提供する簡単な MapElement の作成方法を示したものです。まず、一部のコンストラクタで使われる配列をいくつか設定します。次に、ImageElement を追加します。最後に、MapElement を作成します。

```
  int[] xcoords = {353, 475, 353};
  int[] ycoords = {116, 163, 163};
  int ncoords = 3;
  int[] rectcoords = {353, 33, 475, 53};

  hp.getBody()
    // 新しい ImageElement を作成し、その setUseMap() メソッドを呼び出す
    .addElement(
       new ImageElement(
           "http://www.weblogic.com/images/ipscreen.gif")
              .setAlign(AlignType.center)
              .setUseMap("#MyMap", true))
    // 新しい MapElement を作成する
    .addElement(new MapElement("MyMap")
              // AreaElement を追加する。この場合は、
              // 座標として int 型の配列を使う
              .addElement(new AreaElement()
                          .setShape(AreaElement.rectangle)
                          .setHref("htmlimgmap.html#ok")
                          .setCoordinates(rectcoords))
              // このコンストラクタは、座標として
     // 文字列を取る。
              .addElement(new AreaElement(AreaElement.rectangle,
                           "353, 60, 475, 82",
                           "htmlimgmap.html#cancel"))
     // このコンストラクタは、引数として
     // AWT の Rectangle を取る
              .addElement(new AreaElement(new java.awt.Rectangle(
                    353,88,475,100),
                                    "htmlimgmap.html#dns"))
     // このコンストラクタは、引数として
     // AWT の Polygon を取る
              .addElement(
new AreaElement(new java.awt.Polygon(xcoords,
                                     ycoords,
                        ncoords),
                                    "htmlimgmap.html#advanced"))
              .addElement(
   new AreaElement(AreaElement.circle,
                                    "220, 193, 40",
                                    "htmlimgmap.html#circle")));

   hp.output();
```

フレームの使い方

フレーム用のクラス

- weblogic.html.FrameElement
- weblogic.html.FrameSetElement
- weblogic.html.NoFramesElement
- weblogic.html.ScrollType

FrameElement

ページ上のフレームごとに 1 つの FrameElement を使います。このクラスには、フレームのマージン、ターゲット ウィンドウの名前とソース、スクロールの振る舞い、そしてフレームをサイズ変更できるかどうかを設定するためのメソッドがあります。

注意： フレームの noresize 属性を true に設定すると、ページ上で一辺を共有するその他のフレームも自動的に noresize にセットされる点に注意してください。

htmlKona は、スクリプトを使用するための onLoad 属性と onUnload 属性だけでなく、枠の色（bordercolor）とフレーム枠（frameborder）を設定するためのブラウザ拡張機能もサポートしています。

FrameSetElement

FrameSetElement を FrameElement のコンテナとして使用して、フレームを使う HTML ページを設計します。まず新しい FrameSetElement オブジェクトを作り、次にその addElement() メソッドを使って FrameElement を追加します。

各 FrameSetElement は、2 つの FrameElement を含むこともできますし、他の FrameSetElement を含むこともできます。複数の FrameSetElement を使うと、フレームの中にフレームを置くことができます。setCols() と setRows() を使い、FrameSetElement に含まれている 2 つのオブジェクトの間でスクリーン サイズを分割します（ピクセル単位かパーセント単位で）。htmlKona では、bordercolor

（枠の色）、frameborder（フレーム枠）、および border（枠）の各属性のためのブラウザ拡張機能もサポートしています。FrameSetElement 内の要素には、整数インデックスによってアクセスできます。

NoFramesElement

NoFramesElement を使って、ブラウザがフレームを表示できない場合に表示すべきフレーム付きページのコンテンツを設定します。

ScrollType

ScrollType クラスは、htmlKona で提供されるフレームに有効なスクロール属性をカプセル化します。ScrollType を使って、FrameElement のスクロール特性を設定します。FrameElement クラスの setScrolling() メソッドの引数として、ScrollType オブジェクトを使います。

例

この例では、*fs* という FrameSetElement オブジェクトを 1 つ作成し、そのパネルのサイズをページの 30% と 70% に設定します。次に、FrameSetElement の小さい方の側を *Frame1* という FrameElement に設定します。続いて、2 つの FrameElement（*Frame2* と *Frame3*）を含む別の FrameSetElement オブジェクト *fs2* を作成します。FrameSetElement オブジェクト *fs2* 全体を、ページの大きい方の側として、最初の FrameSetElement に追加します。

```
  String base = "/ns-home/frameExample";
  FrameSetElement fs = new FrameSetElement()
                          .setRows("30%,70%")
.setBorderColor(HtmlColor.fuchsia)
.setFrameBorder(true);
  fs1.addElement(new FrameElement().setName("Frame1")
                          .setSource(base + "?name=Frame1")
                 .setScrolling(ScrollType.no));
  FrameSetElement fs2 = new FrameSetElement()
                          .setCols("50%,50%");
  fs2.addElement(new FrameElement().setName("Frame2")
                          .setSource(base + "?name=Frame2"));
  fs2.addElement(new FrameElement().setName("Frame3")
                          .setSource(base + "?name=Frame3"));
  fs1.addElement(fs2);
  hp.setFrameSetElement(fs1);
```

テーブルの使い方

テーブル用のクラス

- weblogic.html.TableElement
- weblogic.html.TableCaptionElement
- weblogic.html.TableHeadingElement
- weblogic.html.TableRowElement
- weblogic.html.TableDataElement

htmlKona の主要なテーブル オブジェクトは、**TableElement** です。まず、TableElement オブジェクトを作成し、その説明文と属性を設定した後、それに表題やデータの行を追加します。

一般に TableElement は **TableRowElement** から成り、また TableRowElement は一般に、**TableDataElement** か、特殊な種類の TableDataElement である **TableHeadingElement** から成ります。TableDataElement はテーブルの単一のセルに相当し、TableDataCell オブジェクトを作ってからその内容も設定することができます。また、TableRowElement と TableDataElement は、文字列または他の HtmlElement を使って作成することもできます。たとえば、ImageElement を使って、テーブルのセルの中に画像を表示することができます。テーブルの説明文を設定するには、**TableCaptionElement** を使います。

また、dbKona DataSet を使って TableElement を作成することもできます。htmlKona を使って、dbKona DataSet 内のデータを表示すると便利です。DataSet に関連付けられている Schema オブジェクトが、TableElement オブジェクトの作成時に TableElement の構造を自動的に与えることができるからです。同様に、dbKona DataSet 内の単一の Record を使って TableRowElement オブジェクトを作成することができます。

TableElement クラスには、行とセルの追加、取得、および設定のためのメソッドがあります。TableElement 内の行とセルには、インデックス位置によってアクセス可能です。

また、TableElement クラスには、border、caption、cellpadding、cellspacing、および width などの属性を設定するためのメソッドもあります。htmlKona では、背景色属性や画像属性といったテーブル用のブラウザ拡張機能と、Mosaic の BORDERSTYLE 属性（および、weblogic.html.BorderstyleType クラスから提供

されるカプセル化定数）をサポートしています。また、HTML3.5 での採用を提案された FRAME 属性と RULES 属性もサポートしています。これらの属性を設定するには、weblogic.html.FrameType クラスと weblogic.html.RulesType クラスから提供されるカプセル化定数を使います。また、AlignType オブジェクトを使って、テーブル要素の位置合わせを設定します。

例

この例では、テーブルを 2 つ作成して、一方のテーブルをもう一方の中に入れ子にする方法を示します。任意の HtmlElement を付加することによって TableRowElement オブジェクトを作る場合もあれば、TableDataElement を付加する場合もある点に注意してください。TableRowElement.addElement() の引数には任意の HtmlElement を使うことができますが、位置合わせや背景色など、何らかの特性をテーブルのセルに設定したい場合には、セルを TableDataElement として追加し、そのクラス内のメソッドを使ってセルの特性を設定しなければなりません。

```
// テーブルを作成し、属性と説明文を設定する
// それらは、そのテーブルの最下部にイタリック体の
// フォントで表示される
TableElement tab = new TableElement()
  .setBorder(1)
  .setCellPadding(5)
  .setWidth("50%")
  .setCaption(new TableCaptionElement(
         new ItalicElement("Fall Registration Schedule"))
        .setAlign(AlignType.bottom));
// テーブル行を追加し、それにテーブルの表題を入力する。
// テーブルの最初の表題は、AnchorElement として作成する
// ことによって内部アンカーとして設定する点に注意。
// また、2 列目の表題には異なる背景色を
// 設定する
tab.addElement(new TableRowElement()
       .addElement(
   new TableHeadingElement(new AnchorElement(
                            AnchorType.name,
                                    "topic",
                                    "Topic")))
          .addElement(new TableHeadingElement("Scheduled")
                .setBgColor("#A93B6F")));
// 2 行目を追加する。1 列目ではメソッドを呼び出して
// ID を取得し、2 列目では
// 入れ子テーブルを作成する。この
// 入れ子テーブルは、TableDataElement の setVAlign()
// メソッドを呼び出すことで、セルの最上部に配置する
tab.addElement(new TableRowElement()
```

```
            .addElement(new TableDataElement(reg.getID())
        .setVAlign(AlignType.top))
    // この入れ子テーブルは、dbKona DataSet を
    // 取得するメソッドの出力である
            .addElement(new TableDataElement(
        new TableElement(reg.getMain())
                        .setCaption(new BoldElement("N1"))
                        .setBorder(1))
                        .setVAlign(AlignType.top)));
// 単純な文字列を含む行を追加する
tab.addElement(new TableRowElement()
            .addElement("RegID 609E")
   .addElement("N2 - 10AM MWF"));
// 最後に、中央に位置する要素として
// テーブルを HtmlPage に追加する
hp.getBody()
     .addElement(new CenteredElement(tab));
```

フォームの設定

フォーム用のクラス

- weblogic.html.FormElement
- weblogic.html.InputElement（これは、クラス weblogic.html.FieldType を使う）
- weblogic.html.SelectElement（これは、クラス weblogic.html.OptionElement を使う）
- weblogic.html.TextAreaElement（これは、クラス weblogic.html.WrapType を使う）

htmlKona でフォームを作成するには、FormElement オブジェクトを作成し、それに InputElement、SelectElement、あるいは TextAreaElement を追加します。次に、HtmlPage.addElement() を使って、この FormElement をページに追加します。

HTML と同じように、FormElement を TableElement の中に入れ子にできます。このため、フォームのレイアウトが簡単になります。TableElement を使ってフォームのレイアウトを設定する場合には、テーブル属性を設定するときに、FormElement のコンテンツに対してではなく TableRowElement または TableDataElement に対して操作を行っていることを確かめてください（これについては、後述の例でさらに説明します）。

InputElement のタイプは、コンストラクタ内で weblogic.html.FieldType オブジェクトを使って設定されます。FieldType クラスは、チェック ボックス、テキスト入力領域、ラジオ ボタン、送信ボタンなどを含め、有効なタイプの HTML 入力要素をすべてカプセル化しています。このクラスは、以下のものもサポートしています。

- Button（ボタン）
- Checkbox（チェック ボックス）
- File（ファイル）
- Hidden（隠し要素）
- Image（画像 : 送信ボタンに画像を使うため）
- Password（パスワード）
- Radio（ラジオ ボタン）
- Reset（リセット ボタン）
- Submit（送信ボタン）
- Text（テキスト入力領域）

SelectElement はドロップダウン タイプまたはスクロール タイプの選択項目ボックスをフォーム上に作成するもので、OptionElement から成ります。まず SelectElement オブジェクトを作成し、addElement() メソッドを使って選択肢を付加します。選択肢には、文字列か OptionElement を使うことができます。OptionElement クラスのメソッドのうち、どれかを使って属性を設定する必要がある場合には、OptionElement を使います。OptionElement は、表示される文字列のほかに、表示されない値も持っています。

TextAreaElement は、ユーザ入力用の複数行テキスト ボックスです。setContent() メソッドを使うと、ユーザに対して何らかのデフォルト テキストをテキストボックスに表示できます（1 行のテキストボックスの場合には、FieldType.text の InputElement を使います）。setRows() メソッドと setCols() メソッドを使って、テキスト ボックスの高さと幅を設定します。htmlKona は、属性 ONFOCUS、ONBLUR、ONSELECT、ONCHANGE などの拡張機能もサポートしています。

TextAreaElement のテキストラップ特性を設定するには、**WrapType** オブジェクトを使います。WrapType オブジェクトを指定して TextAreaElement.setWrap() メソッドを使い、テキスト ラップをオフ、仮想ラップ、または物理ラップに設定します。

フォーム データの取得

Java 対応サーバに対して htmlKona の FormElement を処理する際には、ユーザが送信ボタンを使って入力した情報、たとえば Common Gateway Interface（CGI）変数などを取り出さなければなりません。ごく簡単に言えば、ユーザがフォーム上で送信ボタンを押すと、新しいフォームを表示したのと同じクラスが再び呼び出されて、フォーム上でユーザが入力した情報を取り出して利用します。一般に、CGI 変数は、FormElement の NAME 属性を参照します。変数の内容は、入力要素のタイプによって決まります。すなわち、textarea 要素は文字列を返し、select 要素は選ばれた選択肢のインデックス位置を返します。

Java Server でも Netscape サーバでも、CGI 変数は Hashtable オブジェクトとして格納され、変数の内容を取り出すには get() メソッドを使います。

以下の例では、Java Server と Netscape サーバの双方に対するサーバサイド Java の実際の使い方を、Java Server と Netscape の実装上の詳細を含めて説明します。

例

この例では、FormElement の作成に必要な手順を示します。さらにこの例では、TableElement を使ってフォーム上のユーザ入力領域を整列させます。この例には、以下に示す 6 つのステップが含まれます。

1. HtmlPage オブジェクトを作って、そのタイトルを設定します。ヘッダ（head）部分に他の属性を追加でき、また HtmlPage.getBodyElement() を呼び出して、ページの本文（body）部分に他の属性を設定できます。
2. FormElement オブジェクトを作成します。
3. TableElement オブジェクトを作成し、フォームのさまざまな部分を構成します。
4. InputElement を含んだ行をこのテーブルに追加します。

5. TableElement と、追加する他のすべての InputElement を FormElement に追加します。

6. 他の HtmlElement と一緒に、FormElement を HtmlPage に追加します。

```
import weblogic.html.*;
import java.io.*;

public class myform {

  public static void main(String argv[])
      throws HtmlException, IOException {

    // ページを作成してそのタイトルを設定する
    HtmlPage hp      = new HtmlPage();
    hp.getHead()
      .addElement(new TitleElement("Creating a Form with
htmlKona"));

    // フォームを作成する
    FormElement form = new FormElement("/servlet/myform",
                                       "POST");

    // テーブルを作成し、それを使ってフォーム要素を構成する
    TableElement tab = new TableElement().setBorder(0);

    // フォームの入力要素とラベルを
    // テーブル内の行として追加する
    tab.addElement(new TableRowElement()
            .addElement(new TableHeadingElement("Your name"))
    .addElement(new TableHeadingElement("Password"))
    .addElement(""));
    tab.addElement(new TableRowElement()
    .addElement(new InputElement("namefield",
                          FieldType.text)
        .setValue("Your name")
        .setSize(20)
.setMaxlen(30))
    .addElement(new InputElement("password",
                         FieldType.password)))
.addElement(new TableRowElement()
    .addElement(new TableDataElement(
                  new InputElement("submit",
                           FieldType.submit)
          .setValue("Check my account"))
.setColspan(2)));

    // TextAreaElement とテーブルをフォームに追加する
    form.addElement(new TextAreaElement("notes",
                                        "Enter your comments")
    .setCols(30))
        .addElement(tab);
```

```
    // フォームをページに追加する
    hp.getBody()
      .addElement(MarkupElement.HorizontalRule)
      .addElement(new HeadingElement("Feedback Form", 2))
      .addElement(form)
      .addElement(MarkupElement.HorizontalRule);
    hp.output();
  }
}
```

ページへのスクリプトの付加

スクリプトで使用するクラス

- weblogic.html.ScriptElement

注意： スクリプト関数は、多くの htmlKona クラス内のメソッドを使って属性 ONBLUR、ONCHANGE、ONLOAD、ONUNLOAD などに設定できる、ページ内のイベント ハンドラから呼び出されます。

ScriptElement の使い方

ScriptElement は、HtmlPage.getHead() メソッドを呼び出した後で、ドキュメントの HEAD 部分に追加することができます。また ScriptElement は、addElement() メソッドを使っていくつかのクラスに追加することもできます。

例

以下の Netscape の例では、GET アクションでフォームの一部として実行される簡単な JavaScript スクリプトを示します（Netscape のサーバサイド Java はパッケージ名をサポートしない点に注意してください）。HtmlPage.getHead() メソッドを呼び出した後に、ScriptElement が追加されます。この例では、ScriptElement を使って式を計算し、答えを返します。

```
import netscape.server.applet.HttpApplet;
import java.io.*;
import weblogic.html.*;

public class adder extends HttpApplet {

  public synchronized void run() throws IOException {
    try {
```

```
        // Netscape のサーバサイド Java クラス用の
        // ルーチン処理
        this.returnNormalResponse("text/html");

        // ページ オブジェクトを作成する
        ServletPage hp = new ServletPage("Example 2d");

        // ユーザ入用の FormElement を作成する
        FormElement form = new FormElement("", "GET");

        // テーブルを使ってフォーム上の要素を整列させ、
        // そのテーブルをフォームに追加する
        TableElement tab = new TableElement();
        form.addElement(tab);

        // テーブルに InputElement を入れる。セルが 3 つある行を
        // 1 つ作成する
        tab.addElement(new TableRowElement()
                .addElement(new InputElement("input")
                            .setValue("9 + 3")
                            .setSize(20)
                            .setMaxlen(20))
                .addElement(new InputElement("button1",
                              FieldType.button)
                            .setValue("Evaluate")
                            .setOnClick("compute(this.form)"))
                 .addElement(new InputElement("answer")
                            .setValue("The answer is: 12")
                            .setSize(20)
                            .setMaxlen(20)));
        // getBodyElement() メソッドを使って
        // BODY タグの属性を設定する
        hp.getBodyElement()
          .setAttribute(BodyElement.bgColor, HtmlColor.white);

        // getBody() を呼び出して、ScriptElement を含む
        // さまざまな要素をページに追加する
        hp.getBody()
          .addElement(MarkupElement.HorizontalRule)
          .addElement(
    new ScriptElement(


              "document.write(\"This is a simple JavaScript "
                      + "expression evaluator.\")"))
          .addElement(MarkupElement.BeginParagraph)
          .addElement(
    new ScriptElement(
              "document.write(\"Enter an expression on the " +
                    "left and click on Evaluate.\")"))</font><font
color=#A93B6F face="Courier New">

// "\n" を使って ScriptElement 内で強制改行をしている点に
// 注意すること。これによって JavaScript が
// きれいに整列する
```

```
        .addElement(new ScriptElement(
    "function compute(form) {\n" +
            "  if (confirm(\"Evaluate Expression \" +
                form.input.value + \" ?\"))\n" +
            "    form.answer.value = \"The answer is: \" +
                eval(form.input.value);\n" +
            "  else\n" +
            "    alert(\"Try again and choose OK.\");\n" +
            "}"))

// 最後に、フォームをページに追加する
        .addElement(form)
        .addElement(MarkupElement.HorizontalRule);

      // HttpApplet の getOutputStream() メソッドの実行結果を
      // 引数として、ServletPage.output() メソッドを呼び出す
      hp.output(getOutputStream());
    }
    catch (Exception e) {
      e.printStackTrace());
    }
  }
}
```

Netscape HttpApplet クラスのマニュアルは、Netscape のサイト（http://developer.netscape.com/viewsource/index_frame.html?content=husted_js）で参照できます。

ページへのアプレットの付加

アプレット用のクラス

- weblogic.html.AppletElement
- weblogic.html.AlignType
- weblogic.html.ParamElement
- weblogic.html.TextFlowElement

AppletElement

AppletElement を使って、アプレットの属性を設定し、そのアプレットを htmlKona ページのどこに配置すべきかを記述します。applet 要素に必要な 3 つのパラメータ、CODE 属性、WIDTH 属性、および HEIGHT 属性は、コンストラクタの中で使われます。また、AppletElement クラスには、アプレットの位置

合わせ、コード ベース（URI）、アプレット名、左右のマージン幅、上下のマージン幅、およびアプレットの代わりに表示されるテキストをそれぞれ設定するためのメソッドもあります。

さらに、htmlKona では、ユーザのディスクにダウンロードする .zip ファイルを指定して、アプレットを迅速にロードできるようにする ARCHIVE ブラウザ拡張機能をサポートしています。.zip ファイルはアプレットの CODEBASE からの相対パスとして検索されること、および .zip ファイルは**圧縮されていてはいけないこと**に注意してください。アプレットに必要で .zip ファイルに含まれていないクラスは、従来の方法で検索されます。

htmlKona はまた、applet タグ内での HTML3.2 TEXTFLOW 要素の使用もサポートしています。これは、アプレットのロードや実行ができない場合に表示される代替テキストを設定するために使われます。HTML 仕様 2.0 以降では、アプレットの代替テキストを設定しなければならない点に注意してください。必要であれば AppletElement.setAlt() メソッドを使用でき、または AppletElement.addElement() を使って TextFlowElement オブジェクトを追加することもできます。後者をお勧めします。

アプレットのパラメータを設定するには、AppletElement.addElement() メソッドを使って、AppletElement に ParamElement を付加します。ParamElement は、一対の名前と値です。AppletElement に付加された ParamElement には、getElementAt() メソッドを使ってインデックス位置によってアクセスできます。パラメータ名では大文字と小文字を区別することに注意してください。

AppletElement オブジェクトへの引数として AlignType オブジェクトを使って、ページ上でのアプレットの位置合わせを設定します。また、AlignType オブジェクトは、他のオブジェクトの位置合わせの設定にも使われます。

例

この例では、JavaSoft から BlinkingText 要素をロードする Java Server スタイルのサーブレットを示します。このクラス ファイルは、配布キットの weblogic/examples/htmlkona/ の中にあります。showException メソッドは、配布キット内の別のクラス（デフォルト）の一部です。

```
package examples.htmlkona;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import weblogic.html.*;
```

```
public class appletexample extends HttpServlet {

  public synchronized void service(ServletRequest req,
                                   ServletResponse res)
    throws IOException
  {
    try {
      // Java Server サーブレット用のルーチン処理
      res.setStatus(ServletResponse.SC_OK);
      res.setContentType("text/html");
      res.writeHeaders();

      // AppletElement を作成する
      AppletElement applet =
        new AppletElement("Blink.class",
                          300,
                  100,
  AlignType.center);
      applet.setCodeBase("http://www.weblogic.com/classes/")
            .addElement(new ParamElement("lbl ",
 "This is the next best thing to sliced bread! "
 + "Toast, toast, toast, butter, jam, toast, "
          + "marmite, toast."))
            .addElement(new ParamElement("speed", "4"));

      // Servletpage を作成し、その属性を設定する
      ServletPage hp = new ServletPage("Blinking Text Applet");
      hp.getBodyElement()
        .setAttribute(BodyElement.bgColor, HtmlColor.white);

      // ページに AppletElement を追加する
      hp.getBody()
        .addElement(new HeadingElement("This is the Blinking " +
                                       "Text Applet from Sun"))
        .addElement(applet)
.addElement(MarkupElement.HorizontalRule);

     hp.output(res.getOutputStream());
    }
    catch (Exception e) {
      defaults.showException(e, res.getOutputStream());
    }


  }
}
```

ページへのファイルの埋め込み

オブジェクトを埋め込むためのクラス

- weblogic.html.EmbedElement

htmlKona は、EMBED HTML 要素をサポートしています。この要素を使うと、任意のオブジェクトを直接ページに埋め込むことができます。埋め込みオブジェクトは、一部のブラウザでアプリケーション固有のプラグインによってサポートされています。また htmlKona は、標準の属性（SRC、HEIGHT、および WIDTH）だけでなく、EmbedElement の HIDDEN、AUTOSTART、NAME の各属性もサポートしています。

埋め込みオブジェクトの取り扱いは、ブラウザの種類によって異なります。この拡張機能を使う場合は、使用するブラウザの動作を理解しておく必要があります。

htmlKona を使った動的データの表示

htmlKona の最も効果的な使い方のひとつとして、動的データを表示できるページの作成が挙げられます。htmlKona と WebLogic のデータベース コネクティビティ製品（dbKona、WebLogic JDBC Server、2 層 jdbcKona ネイティブ JDBC ドライバなど）を使うと、どのような種類の入出力情報でも処理し、情報を動的に提示できる Web アプリケーションを構築することができます。

- dbKona
- WebLogic JDBC オプション
- WebLogic JDBC Server

次に、簡単な Netscape の例を示します。この例は、従業員データベースから取り出され、dbKona QueryDataSet の「qs」に格納されたレコードの中の情報を表示するためのものです。この例では、WebLogic JDBC（JDBC を Java だけで実装したもの）を使い、WebLogic を通じて Oracle データベースに接続します。WebLogic JDBC クライアント、WebLogic JDBC Server、および Oracle データベース間の接続の情報は、defaults クラス内のメソッドによって処理されます。

この例を実行するには、このファイルを編集しディレクトリを再コンパイルする必要があります。以下に示すのは、ここで使われる、defaults クラスのメソッドです。

```
import netscape.server.applet.HttpApplet;

import java.io.*;
import java.util.*;
import xjava.sql.*;
import weblogic.db.xjdbc.*;
import weblogic.common.*;
import weblogic.html.*;

public class defaults {

  public static Connection login(T3Client t3)
      throws Exception
    {

    // T3Client、WebLogic JDBC Server、
    // 接続（2 層接続）のパラメータを
    // 設定する
    Properties dbprops = new Properties();
    dbprops.put("user",               "scott");
    dbprops.put("password",           "tiger");
    dbprops.put("server",             defaults.server());

    // T3Client、WebLogic JDBC Server、
    // および DBMS 間の接続（多層接続）の
    // パラメータを設定する
    Properties t3props = new Properties();
    t3props.put("weblogic.t3",        t3);
    t3props.put("weblogic.t3.dbprops",dbprops);

    // WebLogic Server と DBMS 間の接続用の
    //  2 層 JDBC ドライバのクラス名と URL を
    // 設定する
    t3props.put("weblogic.t3.driver","weblogic.jdbc.oci.Driver");
    t3props.put("weblogic.t3.url",   "jdbc:weblogic:oracle");

    // T3Client、WebLogic Server、および DBMS 間の
    // 接続用の多層（Java 専用）ドライバのクラス名と
    // URL を設定する
    Class.forName("weblogic.jdbc.t3client.Driver")
      .newInstance();
    Connection conn =
      DriverManager.getConnection("jdbc:weblogic:t3client",
                                  t3props);
    return conn;
  }

  // T3Client 要求をリスンする WebLogic Server の
  // URL を指定する
```

```
  public static final String t3clienturl()  {
    return (String)System.getProperty("t3url",
                                      "t3://localhost:7001");
  }

  // データベースへの接続に関する情報を提供する
  public static final String server()  {
    return (String)System.getProperty("server", "DEMO");
  }

  // 例外が発生したときにスタック トレースを
  // HTML ページに出力する
  public static void showException(Exception e,
                                   OutputStream out)
    throws IOException
    {
    ServletPage hp = new ServletPage("An Exception occurred");
    ByteArrayOutputStream ostr = new ByteArrayOutputStream();
    e.printStackTrace(new PrintStream(ostr));
    hp.getBody()
     .addElement(new HeadingElement("Exception occurred:", 2))
     .addElement(new LiteralElement(ostr.toString()));
    hp.output(out);
  }

  // htmlKona ページの生成に使われるデフォルトを設定する
  public static void setPageDefaults() {
    TableCaptionElement.defaultAlign= AlignType.top;
    TableElement.defaultCaption     = new TableCaptionElement("");
    TableElement.defaultBorder      = 4;
    TableElement.defaultCellspacing = 2;
    TableElement.defaultCellpadding = 2;
    TableElement.defaultWidth       = "100%";
  }
}
```

次の例では、defaults クラス内のメソッドを使って、DBMS 接続の情報の取得、ページ デフォルトの設定、および例外の表示を行います。この例では、WebLogic JDBC Client を使って、WebLogic の JDBC Server を通じて DBMS に接続します。WebLogic JDBC はクライアントサイド ライブラリを必要としないので、アプレット内で使うのに適していますが、この例では、それをサーバサイド Java で使います。dbKona は高度に抽象化されており、ベンダ固有のデータ構造に依存しません。このため、dbKona を使うと、どのような SQL データベースからでも非常に一貫した方法でデータを取得できるプログラムを作成できます。defaults クラス内の接続情報を変更しさえすれば、この例を Sybase や Microsoft SQL Server で使えるように変えることができます。

この例では、サーバサイド Java を使ってデータベースに接続し、その情報を表示します。dbKona DataSet を使って、データベースに対するレコードの挿入、修正、および削除を行います。dbKona DataSet は、SQL と QBE（Query-By-Example）の自動生成機能を提供するのです。

```
import netscape.server.applet.HttpApplet;

import java.io.*;
import java.util.*;
import weblogic.db.xjdbc.*;
import weblogic.html.*;
import xjava.sql.*;

public class example5 extends HttpApplet {

  public synchronized void run() throws IOException {
    T3Client   t3   = null;
    Connection conn = null;

    try {
      // Netscape のサーバサイド Java 用のルーチン処理
      this.returnNormalResponse("text/html");

      // 接続は defaults クラス内のメソッドで処理される
      t3 = new T3Client(defaults.t3clienturl());
      t3.connect();
      defaults.setPageDefaults();
      conn = defaults.login(t3);

      // SQL 文を 3 つ作成する
      Statement stmt = conn.createStatement();
     String insert = "insert into emp(empno, ename, job, deptno) " +
                     "values (8000, 'MURPHY', 'SALESMAN', 10)";
      String update = "update emp set ename = 'SMITH', " +
                      "job = 'MANAGER' where empno = 8000";
      String delete = "delete from emp where empno = 8000";

      // 1 つ目の SQL 文を実行して、結果を検証する
      stmt.execute(insert);
      TableDataSet ds1 = new TableDataSet(conn, "emp");
      ds1.where("empno = 8000")
         .fetchRecords();

      // 2 つ目の SQL 文を実行して、結果を検証する
      stmt.execute(update);
      TableDataSet ds2 = new TableDataSet(conn, "emp");
      ds2.where("empno = 8000")
         .fetchRecords();

      // 3 つ目の SQL 文を実行して、結果を検証する
      stmt.execute(delete);
      TableDataSet ds3 = new TableDataSet(conn, "emp");
      ds2.where("empno = 8000")
         .fetchRecords();
```

```
        // ページを作成し、その属性を設定する
        ServletPage hp = new ServletPage("Example 5");
        hp.getBodyElement()
          .setAttribute(BodyElement.bgColor, HtmlColor.white);
        hp.getBody()
          .addElement(MarkupElement.HorizontalLine)
          .addElement(new HeadingElement("INSERT results", 2))
          .addElement(new HeadingElement("Using SQL:", 3))
          .addElement(new LiteralElement(insert))
          .addElement(new LiteralElement(ds1))
          .addElement(MarkupElement.HorizontalLine)
          .addElement(new HeadingElement("UPDATE results", 2))
          .addElement(new HeadingElement("Using SQL:", 3))
          .addElement(new LiteralElement(update))
          .addElement(new LiteralElement(ds2))
          .addElement(MarkupElement.HorizontalLine)
          .addElement(new HeadingElement("DELETE results", 2))
          .addElement(new HeadingElement("Using SQL:", 3))
          .addElement(new LiteralElement(delete))
          .addElement(new LiteralElement(ds3))
          .addElement(MarkupElement.HorizontalRule)
.addElement("Copyright 1996-98, BEA Systems Inc.");

        hp.output(getOutputStream());

        // DataSet を閉じる
        ds1.close();
        ds2.close();
        ds3.close();
      }
      catch (Exception e) {
        defaults.showException(e, getOutputStream());
      }

      // 接続を閉じ、切断する
      finally {
        try {conn.close();}    catch (Exception e) {;}
        try {t3.disconnect();} catch (Exception e) {;}


      }
    }
  }
```

Java 対応サーバでの htmlKona の使い方

現在、サーバサイド Java をサポートしている（すなわち、Java クラス ファイルを実行できる）HTTP サーバはいくつかあります。サーバサイド Java をサポートしている HTTP サーバには、JavaSoft の Java WebServer、Netscape の Enterprise Server と Fast Track Server、Oracle の WebServer、そして WebLogic Server などがあります。

WebLogic は、javax.servlet.http.HttpServlet（または weblogic.html.FormServlet）をスーパークラスとする Java クラス ファイルを、HTTP リクエストへの応答として実行できます。dochome が設定されていれば、WebLogic は HTML ページ、画像、アプレット、アーカイブといった任意のファイルを配信することができ、さらに、別の HTTP サーバへのリクエストをプロキシ（代理）として処理することもできます。

htmlKona クラスは、ほとんどの場合サーバサイド Java として実行されます。ここでは、それを WebLogic での htmlKona の使い方という観点から説明します。つまり、既に htmlKona の .java ファイルをクラス ファイルにコンパイルし、それを weblogic.properties ファイル内に登録するとともに、WebLogic Server ホストの CLASSPATH の中に置いてあるということです。ユーザがそのクラス ファイルを（URL で）要求すると、そのクラス ファイルが実行され、その結果が HTTP リクエストへの応答として返されます。

サーバサイド Java として使うためのクラスを htmlKona で作成するということは、そのクラス ファイルに対するリクエストに応答する HTTP サーバに適した形でクラス ファイルを構成しなければならないということです。WebLogic は Java 標準サーブレット API をサポートしており、htmlKona はそのような用途に最適化されています。つまり、適切なクラスをインポートし、HTTP サーバが想定するリクエスト / 応答パターンに合致するクラスを作成するということです。

Java サーバサイド サーブレットの例

ここで、Java Servlet API を使ってサーバサイド Java 用に実装する方法の概要を示します。WebLogic は、この API に準拠するサーブレット、特に javax.servlet.http.HttpServlet のサブクラスを配信します。WebLogic の Web サイトのフォームと対話型ページはすべて、こういったサーブレットを使っており、それらは WebLogic 上で動作します。

Servlet API 小史 : WebLogic は、Jeeves（バージョン A1.2 および A2）として誕生したとき以来、Java Servlet API をサポートしてきました。Java Servlet API のバージョン 1.0 で、JavaSoft はサーブレットのパッケージ名を変えて、**javax** で始まるようにしました。WebLogic バージョン 2.5 以降で使えるようにサーブレットをアップグレードする場合には、パッケージ名を java.servlet から javax.servlet に変更する必要があります。

また、FormServlet のサブクラスを使っている場合もあります。サーブレットをサポートする Web サーバができるずっと前は、htmlKona は FormServlet というクラスをサポートしていました。Jeeves バージョン A2 がリリースされたとき、Jeeves は FormServlet という HttpServlet のサブクラスをサポートし、その時点で、WebLogic は独自の FormServlet を非推奨にして Javasoft 標準をサポートするようになりました。しかし、Servlet API のバージョン 1.0 では、FormServlet がサポートされなくなりました。そのため、FormServlet の機能を使う既存のクラスを持っている可能性のあるユーザを引き続きサポートするため、FormServlet を復活させて htmlKona API に戻しました。リリース 3.0 では、HttpServlet で利用できたフォーム関連機能の方が支持され、FormServlet は非推奨になりました。

既にクラスの中で FormServlet を使っている場合以外は、新たに着手すべきことはありません。すべての機能は、より一般的な HttpServlet で容易に利用できます（特に、doGet() メソッドと doPost() メソッドを実装してクエリを処理することや、service() メソッドをオーバーライドして、リクエストの種類に応じて doGet() か doPost() のどちらかにリクエストをディスパッチすることもできます。doGet() メソッドと doPost() メソッドは、HttpServlet において no-ops(ノー オペレーション) として実装されています)。そのため、FormServlet クラスは非推奨になりましたが、WebLogic では、下位互換性を保つため、引き続きそれをサポートします。

- javax.servlet.*、javax.servlet.http.*、および weblogic.html.* をインポートします。
- クラスは、javax.servlet.http.HttpServlet を拡張しなければなりません。
- クラスは、HttpServlet.service() メソッド（HttpServletRequest オブジェクトと HttpServletResponse オブジェクトという 2 つの引数を取る）を実装しなければなりません。あるいは、サーバにポストしたいフォーム データを使う場合には、クラスは HttpServlet.doGet() メソッドまたは HttpServlet.doPost() メソッドを実装して、クエリを処理しなければなりません。

- このクラス内の final static int 定数の 1 つを引数に指定して、HttpServletResponse オブジェクトに対して setStatus() メソッドを呼び出して、リクエストの状態を示します。
- ページの content-type（通常 "text/html"）を引数に指定して、ServletResponse オブジェクトに対して setContentType() メソッドを呼び出します。
- weblogic.html.ServletPage を htmlKona キャンバスとして使います。デフォルトのコード セットは「8859_1」です。別の ServletPage コンストラクタを使うことで、他のコード セットを設定することができます。このコンストラクタは、以下に示すように、Codeset オブジェクトを引数に取ります。

```
ServletPage sp = new ServletPage(new Codeset("SJIS"));
```

- ServletPage に対して呼び出す output() メソッドの引数として、HttpServletResponse.getOutputStream() メソッドの実行結果を使います。

```
package examples.htmlkona;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import weblogic.html.*;

public class example1 extends HttpServlet {

  public synchronized void service(HttpServletRequest req,
                                   HttpServletResponse res)
    throws IOException
  {
    // 応答オブジェクトに状態とコンテンツ タイプを設定する
    res.setStatus(HttpServletResponse.SC_OK);
    res.setContentType("text/html");

    // サーブレット ページを作成し、それに文字列を 1 つ付加する
    ServletPage sp = new ServletPage("Example 1");
    sp.getBody().addElement("Hello world!");

    // 応答オブジェクトの出力ストリームを引数として、
    // ページの output メソッドを呼び出す
    sp.output(res.getOutputStream());
  }
}
```

出力テストの簡単な方法

htmlKona は、WebLogic Server などの Java 対応 HTTP サーバとの併用を想定して設計されています。

サーバサイド Java 用のプログラムを開発する場合の欠点の 1 つは、クラスをコンパイルするたびにサーバを再起動しなければならないことです。各クラスは 1 回しか（初めて要求されたとき）ロードされないからです。テストや調整を行っているときは、表示可能な HTML ファイルに htmlKona クラスを出力して、サーバなしでテストをしたい場合があります。

Java コンパイラを使って、コードによって作成される HTML を生成し、出力をファイルに送った後、ブラウザでそのファイルを見てレイアウトや色などをチェックできます（もちろん、サーバがなければ、フォーム内のユーザ入力をテストすることはできません）。htmlKona クラスをファイルに出力するには、以下のようにします。

1. htmlKona コードを記述した .java ファイルを作成します。クラスには public static void main(String[] argv) が含まれていなければならず、また、HtmlPage.output() メソッドには引数を指定してはいけません。
2. コマンドラインから javac を使って、その .java ファイルをコンパイルします。パッケージを使う場合には、calculator.adder や examples/htmlkona 内の任意のクラスの場合のように、完全パッケージ名を指定しなければなりません。javac を実行するシェルに、適切な CLASSPATH を設定したことを確認します。
3. 生成された .class ファイルを Java の CLASSPATH 内のディレクトリにコピーします。または、-d オプションで javac コマンドの出力先を適切なディレクトリに指定します。
4. コマンドラインから java を使って、.class ファイルを呼び出します。引数なしで output() メソッドを呼び出した場合には、HTML は stout に出力されます。java examples.htmlkona.HelloWorld > test.html のように、出力先をファイルに設定することができます。
5. その HTML ファイルをブラウザで表示します。ファイルの冒頭には、通常 HTTP サーバに渡される「Content-type」の行が追加されています。この行は HTTP で配信されるページには現れないので、テストの際には無視してかまいません。