Web アプリケーション フレームワーク IDE ガイド

Sun Java™ Studio Enterprise 7 2004Q4

Sun Microsystems, Inc.
www.sun.com

Part No. 819-1294-10
2004 年 12 月, Revision A

原典: *Web Application Framework IDE Guide*
Part No: 819-0729-10
Revision A

Please
Recycle

目次

はじめに

このガイドでは、開発者向けに Sun™ Java ™ Studio Enterprise 7 2004Q4 開発環境の「エクスプローラ」ウィンドウと、「エクスプローラ」タブからアクセスするときに表示されるアプリケーションのファイルシステム構造のさまざまなビューの概要を説明しています。

お読みになる前に

このマニュアルを読み始める前に、サーブレットや JavaServlet™ ページ (JSP ページ) などの既存の Java 2 Platform, Enterprise Edition (J2EE™ プラットフォーム) Web テクノロジを利用した Web アプリケーションの構築で用いられている概念を理解しておくことを推奨します。

詳しい情報は、以下のリソースから得ることができます。

- Java 2 Platform, Enterprise Edition Specification
  `http://java.sun.com/j2ee/download.html#platformspec`
- J2EE Tutorial
  `http://java.sun.com/j2ee/tutorial`
- Java Servlet Specification バージョン 2.3
  `http://java.sun.com/products/servlet/download.html#specs`
- JavaServer Pages Specification バージョン 1.2
  `http://java.sun.com/products/jsp/download.html#specs`

注 – Sun では、本マニュアルに掲載されている第三者の Web サイトのご利用に関しましては責任はなく、保証するものでもありません。また、これらのサイトあるいはリソースに関する、あるいはこれらのサイト、リソースから利用可能であるコンテンツ、広告、製品、あるいは資料に関しても一切の責任を負いません。Sun は、これらのサイトあるいはリソースに関する、あるいはこれらのサイトから利用可能であるコンテンツ、製品、サービスの利用あるいはそれらのものを信頼することによって、あるいはそれに関連して発生するいかなる損害、損失、申し立てに対する一切の責任を負いません。

マニュアルの構成

第 1 章では、Sun Java Studio Enterprise 7 開発環境 (IDE) の各部の概要、および Web アプリケーションフレームワークを開発するためのビジュアルツールの使用方法を重点的に説明しています。

第 2 章では、「Web アプリケーションフレームワーク」タブの概要を説明し、このタブにある主なノードを 1 つずつ説明しています。

第 3 章では、Web アプリケーションフレームワークアプリケーションで作成する主な Web アプリケーションフレームワークコンポーネントを視覚的に表現するさまざまなノードの概要を説明しています。

第 4 章では、Sun Java Studio Enterprise 7 のツールオプションの概要を説明しています。

書体と記号について

書体または記号*	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例。	.login ファイルを編集します。 ls -a を実行します。 % You have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表します。	% **su** Password:
AaBbCc123 または**ゴシック**	コマンド行の可変部分。実際の名前や値と置き換えてください。	rm *filename* と入力します。 rm **ファイル名** と入力します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照。 この操作ができるのは「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅をこえる場合に、継続を示します。	% **grep '^#define ** **XV_VERSION_STRING'**

* 使用しているブラウザにより、これら設定と異なって表示される場合があります。

関連マニュアル

Java Studio Enterprise のマニュアルとしては、Acrobat Reader (PDF) 形式のマニュアル、チュートリアルと、HTML 形式のリリースノート、オンラインヘルプ、チュートリアルが提供されています。

オンラインで入手可能なマニュアル

ここで紹介しているマニュアルは、docs.sun.com℠ Web サイトおよび Sun Java Studio Enterprise Developers Source ポータルサイト (http://developers.sun.com/jsenterprise) のドキュメントリンクから入手できます。

docs.sun.com Web サイト (http://docs.sun.com) では、インターネットで Sun のマニュアルを参照、印刷、購入することができます。

- 『Sun Java Studio Enterprise 7 2004Q4 リリースノート』- Part No. 819-1302-10

  最新のリリースの変更点や技術的な注意事項を説明しています。

- 『Sun Java Studio Enterprise 7 インストールガイド』(PDF 形式) - Part No. 819-1300-10

  サポートしている各プラットフォームへの Sun Java Studio Enterprise 7 統合開発環境 (IDE) のインストール方法を説明しています。システム要件やアップグレード方法、サーバー情報、コマンド行スイッチ、インストールされるサブディレクトリ、データベースの統合、アップデートセンターの使用方法などの関連情報も記載されています。

- 『J2EE アプリケーションのプログラミング』- Part No. 819-1298-10

  EJB モジュールや Web モジュールを J2EE にアセンブルする方法を説明しています。また、J2EE アプリケーションの配備や実行についても説明しています。

- Web アプリケーションフレームワークのマニュアル (PDF 形式)
  - 『Web アプリケーションフレームワーク コンポーネント作成ガイド』 - Part No. 819-1284-10

    Web アプリケーションフレームワークのコンポーネントアーキテクチャと新しいコンポーネントの設計、作成、配布工程を説明しています。

  - 『Web アプリケーションフレームワーク コンポーネントリファレンスガイド』 - Part No. 819-1286-10

    Web アプリケーションフレームワークライブラリに提供されているコンポーネントを説明しています。

  - 『Web アプリケーションフレームワーク 概要』- Part No. 819-1288-10

    Web アプリケーションフレームワークとその位置づけ、仕組み、他のアプリケーションフレームワークと異なる点を説明しています。

  - 『Web アプリケーションフレームワーク チュートリアル』 - Part No. 819-1290-10

    Web アプリケーションフレームワークを使用して Web アプリケーションを構築する際の仕組みとその手法を紹介しています。

- 『Web アプリケーションフレームワーク 開発ガイド』- Part No. 819-1292-10

  Web アプリケーションフレームワークを使用し、開発するアプリケーションの構成要素として使用可能なアプリケーションコンポーネントの作成および使用の手順と、そのアプリケーションを大部分の J2EE コンテナに配備する方法を説明しています。

- 『Web アプリケーションフレームワーク IDE ガイド』- Part No. 819-1294-10

  Sun Java Studio Enterprise 7 2004Q4 IDE の各部の概要、および Web アプリケーションフレームワークアプリケーションを開発するためのビジュアルツールの使用方法を重点的に説明しています。

- 『Web アプリケーションフレームワーク タグライブラリリファレンス』- Part No. 819-1296-10

  Web アプリケーションフレームワークのタグライブラリを簡単に紹介し、タグライブラリに提供されているタグに対する包括的な参照を示しています。

チュートリアル

Sun Java Studio Enterprise 7 には、IDE の機能を理解する手助けとなるチュートリアルがいくつか用意されています。これらのチュートリアルにある技術、およびコード例は、そのまま、または編集を加えて、実際のアプリケーションの開発に利用することができます。すべてのチュートリアルで、Sun Java System Application Server への配備例が紹介されています。

チュートリアルは、すべて Developers Source ポータルのリンク「Tutorials & Code Camps」から利用可能です。IDE で「ヘルプ」>「コードサンプルとチュートリアル」>「概要」を選択すると、このサイトにアクセスできます。

- 「クイックスタートガイド」は、Sun Java Studio IDE の紹介をしています。チュートリアルは、Sun Java Studio を初めてご使用になる方や、特定の機能について早く知りたい場合は、このガイドから始めてください。これらのチュートリアルは、単純な Web アプリケーションや J2EE アプリケーションの開発方法、Web サービスの生成方法を説明しています。また、UML モデリング、リファクタリングの導入方法についても説明しています。ガイドを終えるための所要時間は数分です。
- 「チュートリアル」は、Sun Java Studio IDE の特定の 1 つの機能に焦点を当てています。ある機能の詳細に関心がある場合は、これらを実行してみてください。例で説明している機能によって、初めからアプリケーションを構築する場合と、提供されたソースファイルを使用して構築する場合があります。チュートリアルは 1 時間以内で完成できます。
- 「概要ビデオ」は、技術の説明がビデオで提供されています。IDE の視覚的な概要や、特定の機能の詳細説明を見ることができます。概要ビデオにかかる時間は数分です。概要ビデオは、任意の個所で開始、終了することもできます。

オンラインヘルプ

Sun Java Studio Enterprise 7 IDE には、オンラインヘルプが用意されています。ヘルプキー (Microsoft Windows 環境では F1 キー、Solaris オペレーティング環境では Help キー) を押すか、「ヘルプ」>「ヘルプ (すべて)」を選択して開くことができます。ヘルプの項目と検索機能が表示されます。

アクセシブルな製品マニュアル

マニュアルは、技術的な補足をすることで、ご不自由なユーザーの方々にとって読みやすい形式のマニュアルを提供しております。アクセシブルなマニュアルは以下の表に示す場所から参照することができます。

マニュアルの種類	アクセシブルな形式と格納場所
マニュアルとチュートリアル	形式: HTML 場所: `http://docs.sun.com`
チュートリアル	形式: HTML 場所: Developers Source ポータル (`http://developers.sun.com/jsenterprise`) のリンク「Examples & Code Camps」
リリースノート	形式: HTML 場所: `http://docs.sun.com`

第1章

Sun Java Studio Enterprise 7 概要

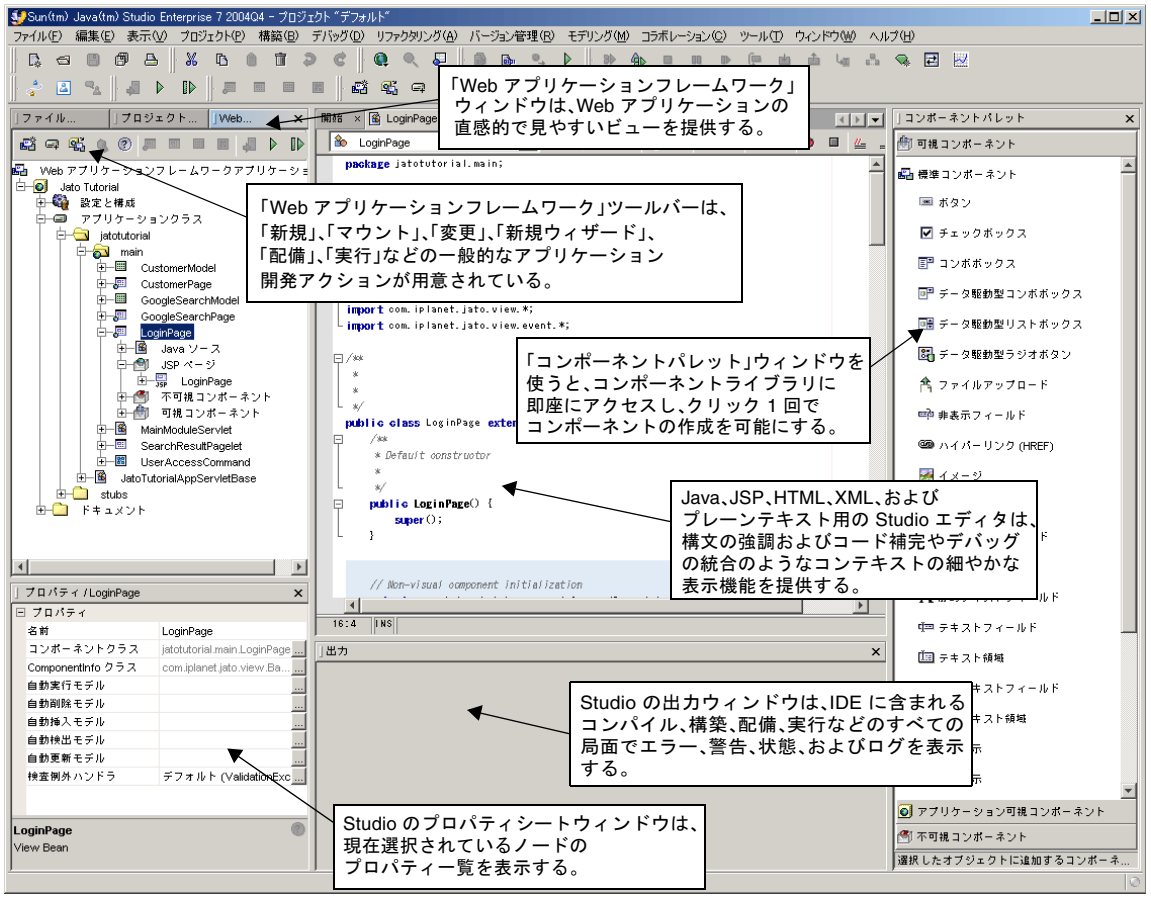
この章では、Sun Java Studio Enterprise 7 開発環境 (以降「IDE」という) の各部の概要を示すとともに、Sun Java Studio の Web アプリケーションフレームワーク (旧称「Sun™ ONE Application Framework」および「JATO」) を使用して Web アプリケーションを開発するためのビジュアルツールの使用方法を重点的に説明します。

このガイドでは、特に IDE の「エクスプローラ」ウィンドウについて説明しています。IDE のこれまでのバージョンでは、これらのウィンドウが「エクスプローラ」ウィンドウのタブを構成していました。IDE の最新バージョンでは、これらのタブは通常の IDE ウィンドウとなり、IDE の「ウィンドウ」メニューにリスト表示されています。このマニュアルで説明するウィンドウおよび IDE 機能には、アプリケーションのファイルシステム構造のビュー、アプリケーション構造、および IDE の環境での実行時統合が含まれます。このマニュアルで取り上げていないタブもあります。

このマニュアルでは、Web アプリケーションフレームワークのツールのさまざまな機能を実際に紹介するための資料として、完成した「JatoTutorial」アプリケーションを使用します。自分の手で「JatoTutorial」を完成していれば、この Web アプリケーションフレームワークアプリケーションの目的を理解していることになります。まだ「JatoTutorial」を完成していない場合は、先に進む前に完成しておくことを推奨します。

NetBeans に基づく Sun Java Studio Enterprise 7 ソフトウェアの基本機能の詳細は、以下の URL にある NetBeans のオンラインマニュアルをご覧ください。
`http://www.netbeans.org/kb/using-netbeans/36/index.html`

次の図は、Web アプリケーションフレームワークを表示しているときの Sun Java Studio Enterprise 7 IDE の全体図です。


「Web アプリケーションフレームワーク」ウィンドウは、Web アプリケーションの直感的で見やすいビューを提供する。
「Web アプリケーションフレームワーク」ツールバーは、「新規」、「マウント」、「変更」、「新規ウィザード」、「配備」、「実行」などの一般的なアプリケーション開発アクションが用意されている。
「コンポーネントパレット」ウィンドウを使うと、コンポーネントライブラリに即座にアクセスし、クリック 1 回でコンポーネントの作成を可能にする。
Java、JSP、HTML、XML、およびプレーンテキスト用の Studio エディタは、構文の強調およびコード補完やデバッグの統合のようなコンテキストの細やかな表示機能を提供する。
Studio の出力ウィンドウは、IDE に含まれるコンパイル、構築、配備、実行などのすべての局面でエラー、警告、状態、およびログを表示する。
Studio のプロパティシートウィンドウは、現在選択されているノードのプロパティー覧を表示する。

Sun Java Studio Enterprise 7 プロジェクト

Sun Java Studio Enterprise 7 のプロジェクトには、IDE「サンドボックス」が用意されています。IDE サンドボックスは、複数の開発環境をそれぞれ独立して扱う手段であり、それぞれの開発業務に合わせてカスタマイズすることができます。初めて Sun Java Studio Enterprise 7 ソフトウェアを実行したときには、Sun Java Studio Enterprise 7 のデフォルトのプロジェクトで作業することになります。

たとえば、無線アプリケーションのために J2ME を使って作業をする必要があると仮定しましょう。その場合は、このプロジェクト専用に、必要なライブラリやファイルシステムをマウントしたり、コンパイラやエディタ、構築オプションを設定したりできます。

ほかにも、CORBA や RMI、リッチ GUI クライアントを使って作業をすることができます。この場合は、他の異なる種類のライブラリやファイルシステムのマウントが必要になったり、非常に異なる構築オプションの設定が必要になったりします。

会社で Web アプリケーションのフレームワークを保守する仕事に従事している場合はどうでしょう。空いている時間に、オープンソースのアプリケーションサーバーに取り組むと仮定しましょう。Sun Java Studio Enterprise 7 では、これらの 2 つのプロジェクトを、「Real Job Project」と「Open Source Project」という、それぞれ独立したプロジェクトにすることができます。

注 – プロジェクトを作成する、または開く時は、必ず Sun Java Studio Enterprise 7 が呼び出したすべてのプロセスを事前に停止し、迅速で無駄のないプロジェクトの切り替えが行われるようにしてください。そのためには、「エクスプローラ」ウィンドウで「実行時」ウィンドウに移動して、「プロセス」ノードを開き、動作中のすべてのプロセスを終了します (プロセスを右クリックして「プロセスを終了」を選択)。

Sun Java Studio Enterprise 7 プロジェクトの詳細は、NetBeans のオンラインマニュアル

(`http://usersguide.netbeans.org/gwd/gwd_project_setup.html#projects`) を参照してください。

Sun Java Studio Enterprise 7 の新規プロジェクトの作成

Sun Java Studio Enterprise 7 の新規プロジェクトを作成するには、「プロジェクト」->「プロジェクトマネージャ」を選択して、「新規」をクリックし、プロンプトが表示されたら、プロジェクト名を入力します。新しい Sun Java Studio Enterprise 7 プロジェクトが作成されて、開きます。プロジェクトの設定は、必要に応じて変更することができます。

Sun Java Studio Enterprise 7 プロジェクトを開く

すでに開いているものとは別の Sun Java Studio Enterprise 7 プロジェクトを開くには、「プロジェクト」->「プロジェクトマネージャ」を選択し、プロジェクトの一覧からプロジェクトを選択して、「開く」ボタンをクリックします。選択したプロジェクトがすでに開いている場合、「開く」ボタンは使用できません。

注意 – 1 つの Sun Java Studio Enterprise 7 プロジェクトに複数の Web アプリケーションフレームワークアプリケーションをマウントする場合は、それらアプリケーションが同じバージョンの Web アプリケーションフレームワーク JAR (`jato.jar`) を使用していることを確認してください。問題は起きないかもしれませんが、曖昧なコンパイルが行われる可能性があります。

Sun Java Studio Enterprise 7 のウィンドウ

Sun Java Studio Enterprise 7 では通常、多くのウィンドウが組み合わせられてタブ付きウィンドウを構成しています。これらのウィンドウのレイアウトや表示のオン/オフは自由に設定できますが、このマニュアルでは、次のウィンドウからなるレイアウトを使用します。

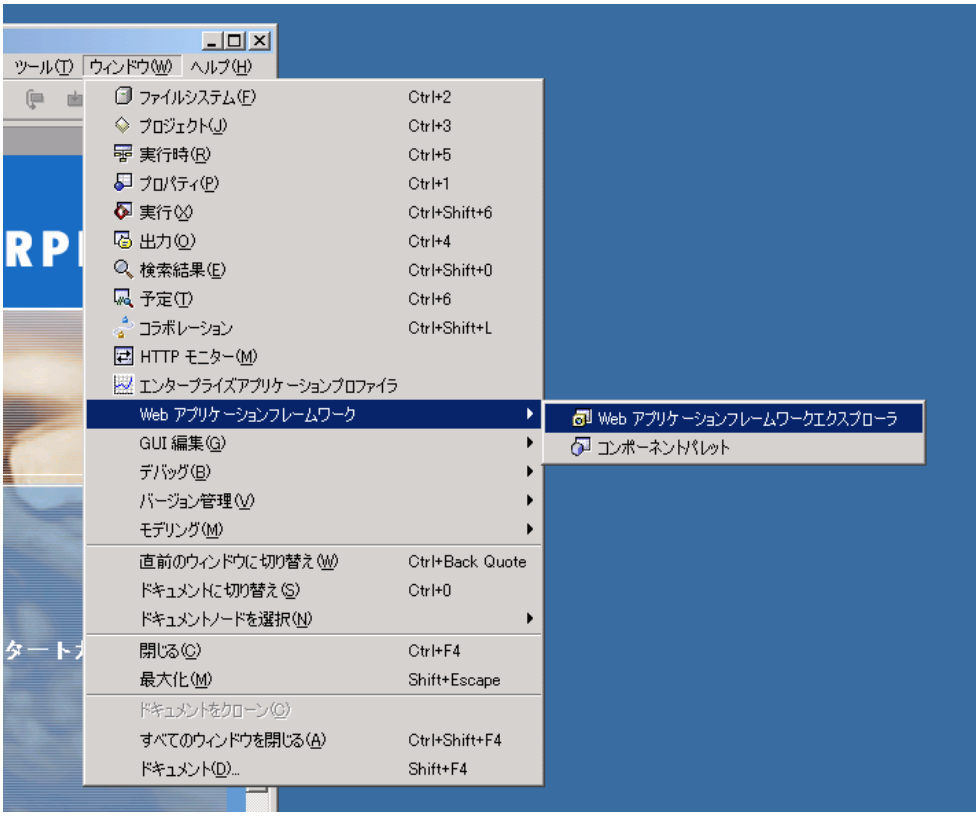
- 「ファイルシステム」
- 「プロジェクト」
- 「Web アプリケーションフレームワークアプリケーション」

これら 3 つのウィンドウは左上に配置されます。

- 「プロパティ」
- 「実行時」

これら 2 つのウィンドウは左下に配置されます。

Sun Java Studio Enterprise 7 ではレイアウトのカスタマイズが非常に自由にできるため、このレイアウトはあくまで一例です。これらのウィンドウは、それぞれ異なる方法で Sun Java Studio Enterprise 環境を表示しています。次の図は、IDE で使用できるすべてのウィンドウの一覧が表示されている「ウィンドウ」メニューです。

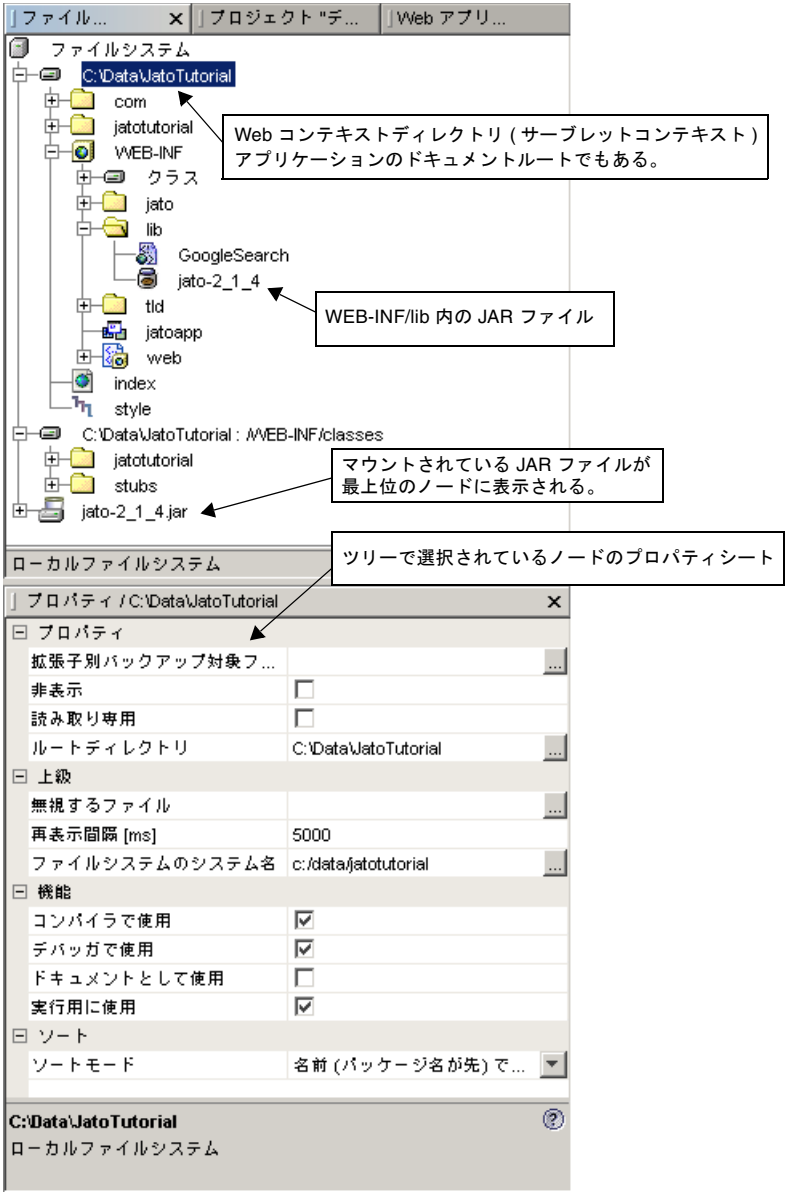


Sun Java Studio Enterprise 7 の「エクスプローラ」ウィンドウの詳細は、下記の URL で NetBeans のオンラインマニュアルを参照してください。

`http://usersguide.netbeans.org/gwd/gwd_project_setup.html#explorer`

「ファイルシステム」ウィンドウ

「エクスプローラ」ウィンドウの「ファイルシステム」ウィンドウは、次の図に示すように、現在マウントされているすべてのディレクトリを表示します。


ファイル...
プロジェクト "デ...
Web アプリ...
ファイルシステム
C:\Data\JatoTutorial
com
jatotutorial
WEB-INF
クラス
jato
lib
GoogleSearch
jato-2_1_4
tld
jatoapp
web
index
style
C:\Data\JatoTutorial : /WEB-INF/classes
jatotutorial
stubs
jato-2_1_4.jar
ローカルファイルシステム
プロパティ / C:\Data\JatoTutorial
プロパティ
拡張子別バックアップ対象フ...
非表示
読み取り専用
ルートディレクトリ
C:\Data\JatoTutorial
上級
無視するファイル
再表示間隔 [ms]
5000
ファイルシステムのシステム名
c:/data/jatotutorial
機能
コンパイラで使用
デバッガで使用
ドキュメントとして使用
実行用に使用
ソート
ソートモード
名前 (パッケージ名が先) で...
C:\Data\JatoTutorial
ローカルファイルシステム
Web コンテキストディレクトリ (サーブレットコンテキスト)
アプリケーションのドキュメントルートでもある。
WEB-INF/lib 内の JAR ファイル
マウントされている JAR ファイルが
最上位のノードに表示される。
ツリーで選択されているノードのプロパティシート

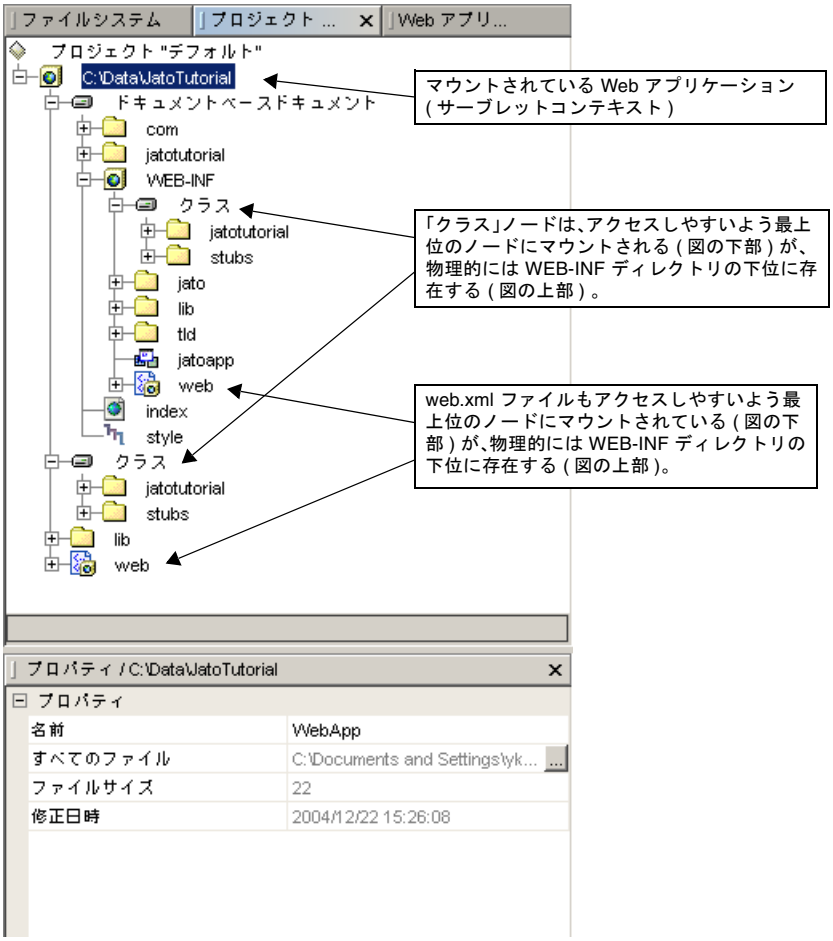
Web アプリケーションフレームワークアプリケーションを新規作成すると、Web コンテキストディレクトリ (別名 : サーブレットコンテキストあるいはベース Web アプリケーションディレクトリ) が自動的にマウントされます。また、Web アプリケーションフレームワークアプリケーションの `WEB-INF/lib` ディレクトリにある JAR ファイルもすべてマウントされます。それら JAR ファイルは `WEB-INF/lib` ディレクトリに物理的に存在するだけですが、最上位のノードに表示されます。

追加のライブラリ (たとえば、`jaxrpc-api.jar` や Web サービスモデル用の他のライブラリなど) を必要とする Web アプリケーションフレームワークコンポーネントを追加したり、アプリケーションの `WEB-INF/lib` ディレクトリに手動で JAR ファイルをコピーしたりすると、それらの新しい JAR ファイルが自動的にマウントされます。Sun Java Studio Enterprise 7 が、新しい JAR ファイルをすぐに認識しない場合があります。これは、読み込みの間隔が Sun Java Studio Enterprise 7 に対する再表示レートの設定に依存しているためです。

JAR ファイルの仮想ビューに加えて、ファイルシステムのビューは、オペレーティングシステムのファイルシステムに見られるような生のディレクトリおよびファイルレイアウトも提供します。Web アプリケーションフレームワークアプリケーションの開発者が、このビューに多くの時間を割く必要はありません。

「プロジェクト」ウィンドウ

Web アプリケーションフレームワークアプリケーションなどの Web アプリケーションの場合、「プロジェクト」ウィンドウには、次の図に示すように、その Web アプリケーションのディレクトリ構造の論理ビュー (平坦化されたビュー) を表示します。



「ファイルシステム」ウィンドウにおける JAR ファイル同様、簡単にアクセスできるよう、中に埋め込まれているノードは、最上位のノードにマウントされます。通常 Web アプリケーションフレームワークアプリケーションの開発者が、このウィンドウを表示する必要はありません。

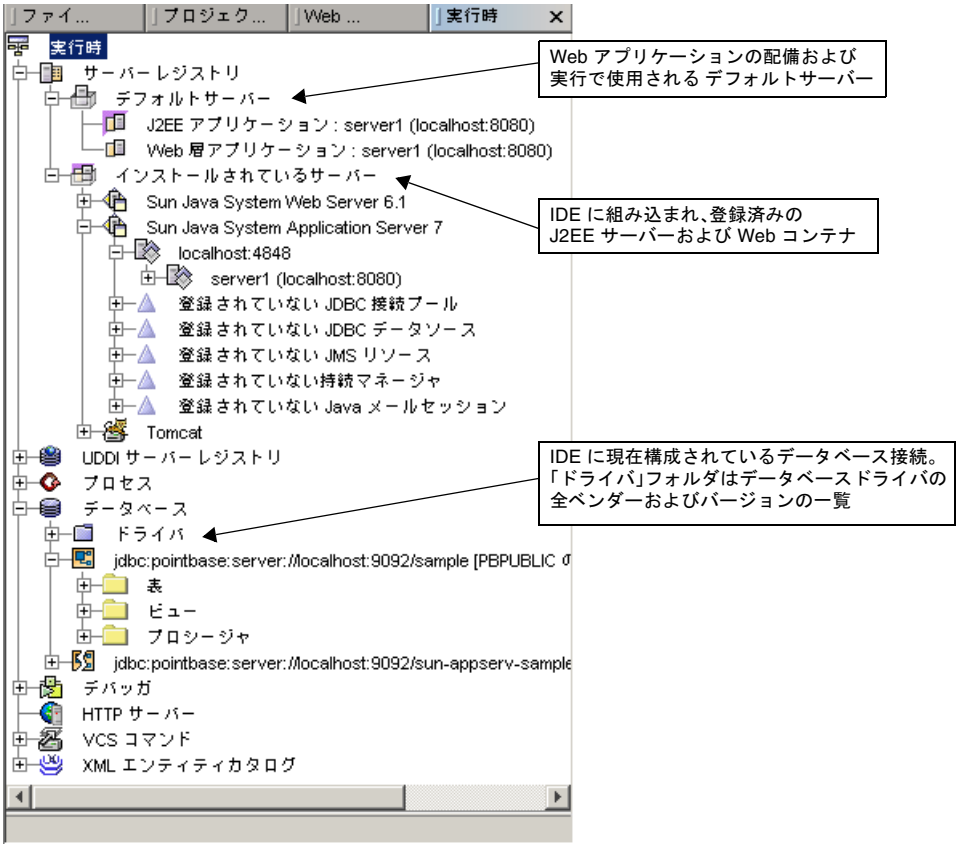
「実行時」ウィンドウ

「実行時」ウィンドウは、Sun Java Studio Enterprise 7 で利用可能なさまざまリソースをツリーノード構造の形式で表示します。

Web アプリケーションや Web アプリケーションフレームワーク Web アプリケーションをはじめようとするアプリケーションの開発中に操作することになるノードには、以下があります。

- 「サーバーレジストリ」
- 「プロセス」
- 「データベース」

次の図は「実行時」ウィンドウを示しています。



「サーバーレジストリ」ノード

「サーバーレジストリ」は、Sun Java Studio Enterprise 7 内から管理、利用することが可能なすべてのサーブレットコンテナの一覧です。Sun Java Studio Enterprise 7 に付属しているサーブレットコンテナには、Tomcat および Sun Java System Application Server の 2 つがあります。このモジュールがデフォルトでインストールされていないときは、Sun Java Studio Enterprise 7 のアップデートセンターにアクセスし、モジュールをダウンロードしてインストールする必要がある場合があります。

「プロセス」ノード

「プロセス」は、Sun Java Studio Enterprise 7 IDE によって起動され、引き続き動作中のプロセスの一覧です。アプリケーションをコンパイルすると、その処理が完了するまで「プロセス」にコンパイルプロセスが表示されます。アプリケーションをテスト実行すると、「プロセス」にサーブレットコンテナプロセスが表示されます。

サーブレットコンテナは、手動で停止されるまで動作中になります。Sun Java Studio Enterprise 7 IDE によって起動されたプロセスを停止するには、「プロセス」リストに移動して、停止するプロセスを右クリックし、「プロセスを終了」を選択します。

Sun Java Studio Enterprise 7 IDE を終了するか、Sun Java Studio Enterprise 7 プロジェクトを切り替える場合は、Sun Java Studio Enterprise 7 IDE によって起動されていたプロセスをすべて停止してください。Sun Java Studio Enterprise 7 のプロジェクトは、Sun Java Studio Enterprise 7 の「プロジェクト」メニューオプションから管理 (「作成」、「削除」、「開く」) します (このマニュアルでは、プロジェクトについては説明していません)。

「データベース」ノード

「データベース」は、Sun Java Studio Enterprise 7 に組み込まれているすべての RDBMS ドライバと接続の一覧です。「ドライバ」フォルダサブノードがあり、データベース接続の一覧が含まれることもあります。

「ドライバ」フォルダサブノードは、ベンダーデータベースの多数のドライババージョンの一覧です。特定のデータベース用のドライバを Sun Java Studio Enterprise 7 が利用できる場合は、「ドライバ」フォルダノードが有効になります。利用できない場合は、無効で、そのアイコンに赤い色の斜線が表示されます。新しいドライバは、「ドライバ」フォルダノードを右クリックし、「ドライバを追加」アクションを選択して、表示されたダイアログボックスに必要なドライバ情報を入力すると追加できます。その場合、Sun Java Studio Enterprise 7 からそのドライバを利用できるようにするには、Sun Java Studio Enterprise 7 の `lib/ext` ディレクトリにドライバを入れておく必要があります。

接続がある場合は、接続されているか、接続が解除されているかのいずれかになります。前者は完全なアイコンで、後者は壊れたアイコンでそれぞれ表示されます。接続するには、接続のノードを右クリックして、「接続」または「名前を変えて接続」ア

クションを選択します。必要な接続情報 (ユーザー名、パスワードなど) の入力が求められます。接続を完了するには、接続先のデータベースが動作している必要があります。このため、必ずデータベースサーバーが起動していてアクセス可能な状態にしておいてください。接続を解除するには、接続を右クリックして「接続を解除」アクションを選択します。

新しい接続は、「データベース」ノードを右クリックして、「接続を追加」アクションを選択することによって追加できます。データベースの種類や場所、接続情報の入力が求められます。この項 (「データベース」ノード) で前述しているように、データベースへの接続を作成するにあたっては、必ず適切なドライバが Sun Java Studio Enterprise 7 から利用できるようにしてください。

「Web アプリケーションフレームワークアプリケーション」ウィンドウ

「Web アプリケーションフレームワークアプリケーション」ウィンドウは、Web アプリケーションの分かりやすいビューを提供します。このウィンドウでは、1 つ以上の Web アプリケーションフレームワークアプリケーションをマウントできます。Web アプリケーションフレームワークアプリケーションをマウントすると、そのアプリケーションが、「ファイルシステム」ウィンドウにファイルシステムとして、また「プロジェクト」ウィンドウに Web モジュールとしてもマウントされます。マウントされた各 Web アプリケーションフレームワークアプリケーション内には、最上位のノードとして、以下に示す 3 つのノードが存在します。

- 「設定と構成」
- 「アプリケーションクラス」
- 「ドキュメント」

これらの各ノードについては、Web アプリケーションフレームワークアプリケーションの作成とマウントに関する詳細を説明している第 2 章「「Web アプリケーションフレームワークアプリケーション」ウィンドウの概要」で簡単に説明します。

第2章

「Web アプリケーションフレームワークアプリケーション」ウィンドウの概要

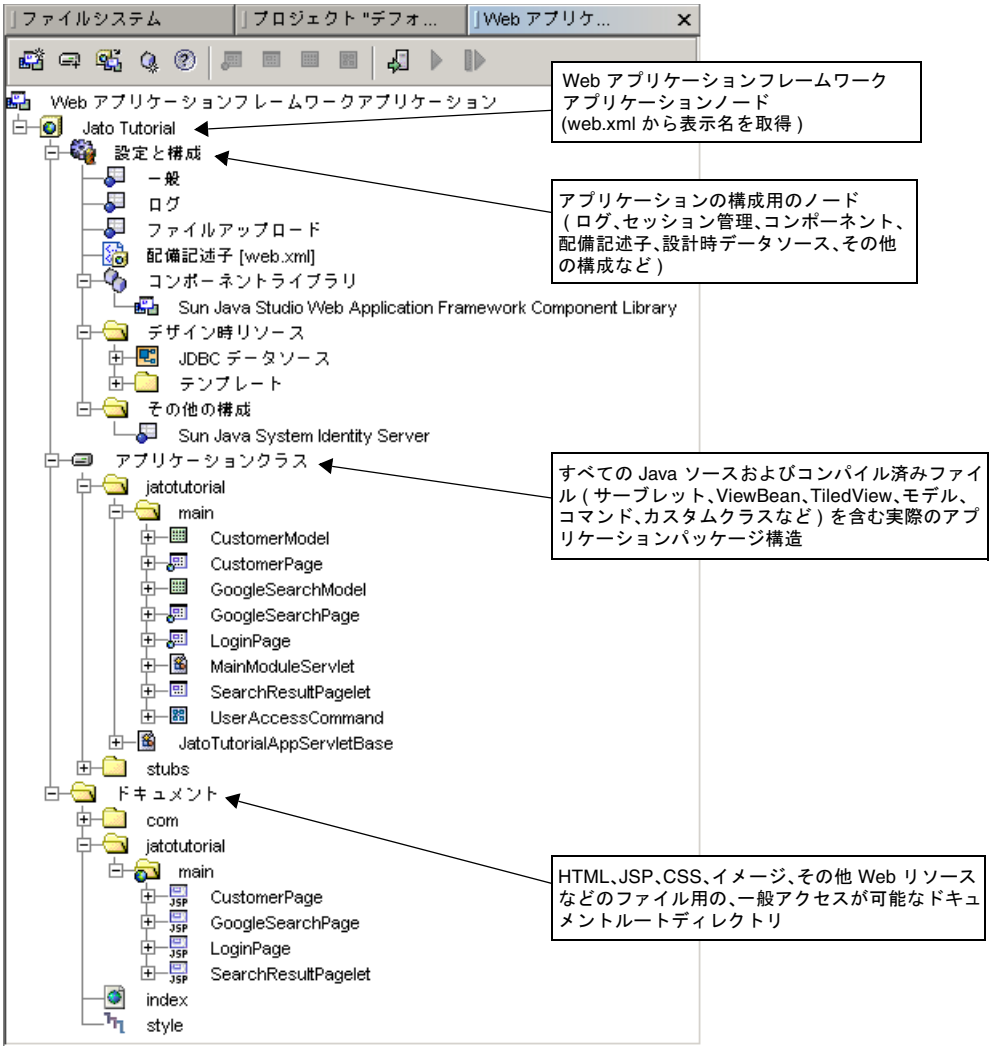
この章では、「Web アプリケーションフレームワーク」ウィンドウの概要を説明し、このウィンドウにある主なノードをそれぞれに説明します。

「Web アプリケーションフレームワークアプリケーション」ウィンドウは、論理的で直感的な方法で Web アプリケーションフレームワークアプリケーションを表示します。

最上位のアプリケーションノードの下に、主要ノードとして以下の 3 つのノードがあります。

- 「設定と構成」
- 「アプリケーションクラス」
- 「ドキュメント」

次の図は、「Web アプリケーションフレームワークアプリケーション」ウィンドウとその 3 つの主要ノードを示しています。



開発者は、作業時間の多くを「アプリケーションクラス」ノードで費やすことになります。Web アプリケーションフレームワークコンポーネントを選択して、プロパティを設定し、イベントやカスタムコードを追加します。このマニュアルでは、これらのノードを 1 つずつ詳しく説明します。

Web アプリケーションフレームワークアプリケーションのルートノード

「Web アプリケーションフレームワークアプリケーション」ウィンドウの一番上にあるノードは「アンカー」ノードであり、この下に 1 つまたは、複数の Web アプリケーションフレームワークアプリケーションをマウントします。マウントされた複数の Web アプリケーションフレームワークアプリケーションは、それぞれ独立したアプリケーションとみなされ、独立した WAR ファイルとして配備されます。

Web アプリケーションフレームワークアプリケーションのルートノードのコンテキストメニューコマンド

Web アプリケーションフレームワークアプリケーションのルートノードを選択して、マウスの右ボタンでクリックすると、以下に説明するメニュー項目を含むコンテキストメニューが表示されます。

▼ [アクション]「新規 Web アプリケーションフレームワークアプリケーション」

Web アプリケーションフレームワークアプリケーションを新規作成するには、Sun Java Studio Enterprise 7 メニューオプションの「ファイル」-「新規」を選択するか、ツールバーの「新規」ボタンをクリックします (このボタンは、通常、Sun Java Studio Enterprise 7 の左上の 1 つ目のアイコンで、「ファイル」メニューオプションのすぐ下に位置しています)。「テンプレートを選択」パネルが表示されます。この「テンプレートを選択」パネルには、Sun Java Studio Enterprise 7 に用意されているテンプレートウィザードがすべて含まれています。

Web アプリケーションフレームワークのウィザードは、「Web アプリケーションフレームワーク」ノードにあります。このノードを開いて、「アプリケーション」を選択します。もう一つ、「Web アプリケーションフレームワーク」ウィンドウの最上部にあるツールバーの「新規 Web アプリケーションフレームワークアプリケーション」ボタンをクリックする方法もあります。このツールバーにあるボタンの一部が、Sun Java Studio Enterprise 7 のメインツールバー (メニューオプションのすぐ下にあるツールバー) に表示されることもあります。これらの方法のどれを使用しても、同じウィザードが呼び出されます。Web アプリケーションフレームワークアプリケーションを新規作成するということは、事実上、新しいアプリケーションをマウントするということです。

▼ [アクション]「Web アプリケーションフレームワークアプリケーションをマウント」

以前に作成した Web アプリケーションフレームワークアプリケーション (バージョン 2.x) が存在していて、初めて Sun Java Studio Enterprise 7 にマウントする場合は、「新規 Web アプリケーションフレームワークアプリケーション」ツールバーボタンの右側にある「Web アプリケーションフレームワークアプリケーションをマウント」ボタンをクリックします。

この他、Sun Java Studio Enterprise 7 の「ファイル」メニューを使用するか、「Web アプリケーションフレームワークアプリケーション」ウィンドウのルートノードで右クリックする方法もあります。このノードは、「Web アプリケーションフレームワークアプリケーション」と呼ばれます。これらのどの方法を使用しても、ファイルシステムのブラウザ/選択ダイアログが開きます。このダイアログを使用して、コンピュータまたはネットワーク上の目的の Web アプリケーションフレームワークアプリケーションがある場所に移動することができます。Web アプリケーションフレームワークアプリケーションのアイコン (重なり合う 3 つの長方形) とともに、Web アプリケーションフレームワークアプリケーションのルートにあるフォルダが表示されます。目的の Web アプリケーションフレームワークアプリケーションを選択し、ダイアログにある「マウント」ボタンをクリックしてください。選択した Web アプリケーションフレームワークアプリケーションが Sun Java Studio Enterprise 7 にマウントされます。

▼ [アクション]「コンポーネントをダウンロード」

現在、このアクションはブラウザウィンドウに「Web アプリケーションフレームワーク」Web ページを開きます。このアクションは、Web アプリケーションフレームワークアプリケーション用にダウンロード可能な Sun 以外の Web アプリケーションフレームワークコンポーネントのリポジトリとなる Web サイトを対象にしています。ただし、こうしたリポジトリはまだ存在しません。Web アプリケーションフレームワークコンポーネントの作成についての詳細は、『Web アプリケーションフレームワーク コンポーネント作成ガイド』をご覧ください。

「Web アプリケーションフレームワークアプリケーション」ノード

Web アプリケーションフレームワークアプリケーションのルートノードのすぐ下には、マウントされている Web アプリケーションフレームワークアプリケーションのノードが 1 つまたは複数存在することになります (ただし、マウントされていない場合ノードは存在しない)。マウントされている各 Web アプリケーションフレームワークアプリケーションのノードは、中央に緑色の地球の入った黄色い立方体で表されます。ノード名は、それぞれの Web アプリケーションの「表示名」になります。表示名は、Web アプリケーションの `WEB-INF` ディレクトリに存在する配備記述子ファイル (`web.xml`) に実際に含まれる要素エントリです。表示名は、ファイルシステムに

おける実際のディレクトリ名とは完全に異なることがあります。このディレクトリ名は、Web アプリケーションの「Web コンテキスト名」といい、「Web ドキュメントのルート」ディレクトリでもあります。

Web アプリケーションフレームワークアプリケーションノードのコンテキストメニューコマンド

このノードには、コンテキストメニューコマンドがいくつかあります。これらアクションの一部を使用して、Web アプリケーションフレームワークアプリケーションを実行する準備をします。

▼ [アクション]「実行」

「実行」アクションは、Web アプリケーションフレームワークアプリケーション内の Web アプリケーションフレームワークページコンポーネントを直接には実行しないため、普通、この種のノードに「実行」アクションを使用することはありません。

「実行」アクションはブラウザウィンドウを起動し、Web アプリケーションのドキュメントのルートにある `index.html` ファイルを読み込もうとします。この `index.html` という名前のファイルが存在しない場合は、Web アプリケーションの `WEB-INF` ディレクトリに存在する配備記述子ファイル (`web.xml`) に宣言されている開始ファイルを読み込みます。

Web アプリケーションフレームワークアプリケーションを作成すると、デフォルトの開始ファイルが作成されます。これは、標準的な開始メッセージの入った簡単な HTML ファイルです。このデフォルトの開始ページの名前は `index.html` で、Web アプリケーションの「ドキュメント」フォルダにあります。このファイルは、カスタマイズすることができますが、Web アプリケーションフレームワークアプリケーションの必須コンポーネントではありません。

ドキュメントのルートディレクトリに `index.html` ファイルがなく、かつ配備記述子ファイルに開始ファイルが宣言されていない場合は、ブラウザウィンドウにドキュメントのルートの標準的なディレクトリ一覧が表示されます。

`index.html` または開始ページは、設定で、Web アプリケーションフレームワークアプリケーションに含まれる特定の Web アプリケーションフレームワークページコンポーネント (ログインページ、メインメニューページなど) にリダイレクトすることができます。

▼ [アクション]「実行 (強制再読み込み)」

このアクションは、ターゲットサーバーにアプリケーションを強制的に再配備し、サーバーを再起動することを除けば、「実行」アクションと同じ動作をします。これは、メモリー上にある Web アプリケーションのコンポーネントを使用しないで、変

更内容が確実に反映されるようにするためです。アプリケーションをコンパイルする必要がある場合は、配備、実行する前にコンパイル (すべてをコンパイル) されます。

▼ [アクション]「配備」

このアクションは、その実行先となるターゲットサーバーに Web アプリケーションを配備します。このアクションは、Web アプリケーションに変更を加えてテスト実行する場合に使用してください。アプリケーションをコンパイルする必要がある場合は、配備する前にコンパイル (すべてをコンパイル) されます。

▼ [アクション]「コンポーネントライブラリを追加」

独自の Web アプリケーションフレームワークコンポーネントライブラリを作成するか、Sun 以外のそうしたコンポーネントを入手した場合は、このアクションを使用して、そのファイルシステムに移動し、選択することができます。選択したコンポーネントライブラリは、アプリケーションの `WEB-INF/lib` ディレクトリにコピーされ、IDE によって自動的にマウントされます。このライブラリは、Web アプリケーションフレームワークコンポーネントライブラリである必要があります。このアクションは、そのようなライブラリ宣言を含む構成ファイル (`complib.xml`) が JAR ファイルに存在することを前提にしています。コンポーネントライブラリの作成についての詳細は、『Web アプリケーションフレームワーク コンポーネント作成ガイド』をご覧ください。

Sun Java Studio Enterprise 7 が新しいライブラリファイルの追加を認識するまでに、多少時間がかかることがあります。この時間間隔は、「ファイルシステム」ウィンドウの「ファイルシステム」ルートノードのプロパティの 1 つで設定されています。必要に応じて、もっと短い間隔に設定変更することができますが、IDE のパフォーマンスの妨げになるため、あまり短い間隔にはしないでください。新規追加するファイルの親になるフォルダノード (この場合は `WEB-INF/lib`) を右クリックし、「フォルダを再表示」アクションを選択することによって、Sun Java Studio Enterprise 7 に強制的にフォルダ状態を再表示させることもできます。

▼ [アクション]「コンパイル」

このアクションは、現在のディレクトリ内の新しいファイルまたは前回コンパイル後に変更されたファイルをコンパイルします。最新かどうかは、コンパイルのソース (`.java`) と生成ファイル (`.class`) のタイムスタンプを比較することによって調べます。サブフォルダ内のファイルはコンパイルされません。

▼ [アクション]「すべてをコンパイル」

このアクションは、サブフォルダ内のファイルを含めて、新しいファイルまたは前回コンパイル後に変更されたファイルをコンパイルします。

▼ [アクション]「構築」

このアクションはフォルダ内の .class ファイルを削除し、ソースファイルを再コンパイルします。サブフォルダ内の .class ファイルを削除することはなく、ソースファイルのコンパイルも行われません。

▼ [アクション]「すべてを構築」

このアクションは、フォルダとそのサブフォルダ内のすべての .class ファイルを削除し、すべてのファイルをコンパイルします。

▼ 「アクション」「モジュールを識別」

このアクションは、web.xml ファイルを検索して、特定のパッケージをモジュールパッケージとして識別する要素エントリを検出します。これは、モジュールを識別するための再表示アクションです。

▼ [アクション]「名前を変更」

このアクションは、Web アプリケーションの「表示名」を変更します。表示名は、アプリケーションの WEB-INF ディレクトリに存在する配備記述子ファイル (web.xml) に実際に含まれる要素エントリです。

▼ [アクション]「WAR ファイルをエクスポート」

このアクションは、指定されたファイル名でアプリケーションをパッケージ化し、選択された場所にコピーします。このアクションは、Sun Java Studio Enterprise 7 の外部にあるサーブレットコンテナでアプリケーションを実行できるよう、配備可能な WAR ファイルを作成する必要がある場合に使用してください。本稼働配備したり、別のコンテナでテスト実行したり、テストのために別の開発者に送付したりする場合です。

▼ [アクション]「アプリケーションをマウント解除」

このアクションは、Sun Java Studio Enterprise 7 環境 (「ファイルシステム」タブ、「プロジェクト」タブ、「Web アプリケーションフレームワークアプリケーション」タブ) から Web アプリケーションフレームワーク アプリケーションをマウント解除 (削除) します。物理的にディスクから削除されるわけではありません。

▼ [アクション]「ツール」

このアクションを使い Sun Java Studio Enterprise 7 のツールにアクセスすることができます (このマニュアルでは、それらツールについては説明していません)。

▼ [アクション]「プロパティ」

このアクションは、新しいウィンドウに、選択されているノードのプロパティシートを表示します。

Web アプリケーションフレームワークアプリケーションノードのプロパティ

▼ [プロパティ]「コンテンツの言語」

Web アプリケーションで実行するページに対するデフォルトのコンテンツの種類です。特定のページに対するコンテンツの種類は、必要に応じてプログラムから変更できます。

▼ [プロパティ]「コンテキストルート」

サーブレットのコンテキスト、すなわち、URL の `server:port` の直後の要素です。

例 : `http://`<server:port>/<contextroot>

このコンテキストルートは、Web アプリケーションフレームワークアプリケーションウィザードで「web コンテキスト名」に指定された値に置き換えられます。たいていの場合、このプロパティを変更する必要はありません。

▼ [プロパティ]「名前」

Web アプリケーションの「表示名」です。表示名は、Web アプリケーションの `WEB-INF` ディレクトリに存在する配備記述子ファイル (`web.xml`) に実際に含まれる要素エントリです。

▼ [プロパティ]「テンプレート」

このノードがテンプレートであるかどうかは、このプロパティによって決まります (「True」/「False」)。

▼ [プロパティ]「Web モジュールグループ」

このプロパティは、Web アプリケーションフレームワークアプリケーションが属している Web モジュールグループファイルのパスを示します。Web モジュールグループの作成の詳細は、IDE のオンラインヘルプを参照してください。

▼ [プロパティ]「追加ファイル」

「WAR ファイルをエクスポート」アクションで使用されるプロパティです。生成される WAR ファイルに含める追加ファイル (Web アプリケーションのディレクトリ構造にないファイル) を指定することができます。

▼ [プロパティ]「フィルタ」

「WAR ファイルをエクスポート」アクションで使用されるプロパティで、生成される WAR ファイルから特定の種類のファイルを除外するのに使用することができます。アプリケーションのソースコードが本稼働のサーバーに公開されることのないよう、Java ソースファイル (`.java`) は WAR ファイルから除外するのが一般的です。

▼ [プロパティ]「デバッガ」

「ツール」->「オプション」メニューオプションで「オプション」ウィンドウを開きます。このウィンドウの「デバッグと実行」ノードの下にデバッガの種類が一覧表示されます。このプロパティには、一覧表示されるデバッガの種類を指定することができます。必要な場合は、現在の種類に基づいて独自のデバッガの種類を作成することができます。

▼ [プロパティ]「実行方法」

「ツール」->「オプション」メニューオプションで「オプション」ウィンドウを開きます。このウィンドウの「デバッグと実行」ノードの下に実行方法が一覧表示されます。このプロパティには、一覧表示される実行方法を指定することができます。必要な場合は、現在の方法に基づいて独自の実行方法を作成することができます。

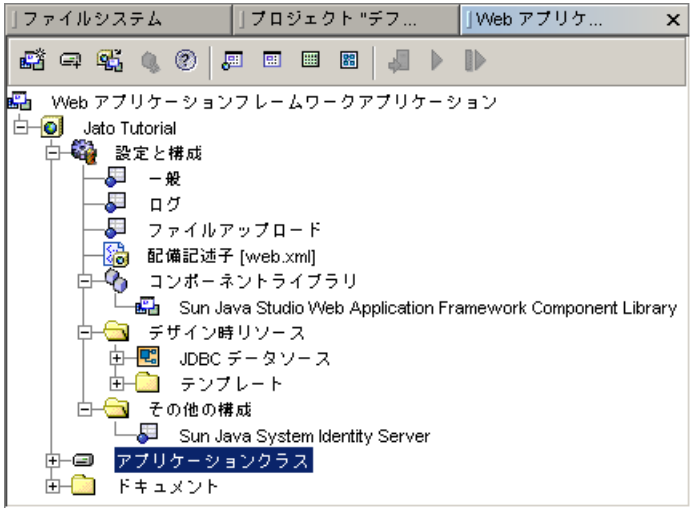
▼ [プロパティ]「ターゲットサーバー」

Web アプリケーションの実行を処理するサーブレットコンテナは、このプロパティによって決まります。デフォルトは 「デフォルト Web サーバー」です。デフォルト Web サーバーは、「サーバーレジストリ」ノードの下の「実行時」ウィンドウで設定します。選択できるのは、Sun Java Studio Enterprise 7 に登録されているサーバーだけです。すべての Web アプリケーションに対して一括で変更をせずに、特定の 1 つの Web アプリケーションの実行のみを処理するサーバーを変更することもできます。

「設定と構成」ノード

「設定と構成」ノードには、Web アプリケーションフレームワークアプリケーションの設計時および実行時環境を構成するための多数のリソースがあります。

次の図は「設定と構成」ノードを示しています。



「設定と構成」ノードのコンテキストメニューコマンド

▼ [なし]

「設定と構成」ノードのプロパティ

▼ [なし]

「設定と構成」ノードのサブノード

「設定と構成」ノードには、以下の 7 つのサブノードがあります。

- 「一般」
- 「ファイルアップロード」
- 「ログ」
- 「配備記述子」
- 「コンポーネントライブラリ」
- 「デザイン時リソース」
- 「その他の構成」

「一般」ノード

「一般」ノードは、補足的な Web アプリケーション構成用のノードです。

「一般」ノードのコンテキストメニューコマンド

▼ [なし]

「一般」ノードのプロパティ

現在、「一般」には、アプリケーションの実行時動作を制御する 2 つのプロパティがあります。

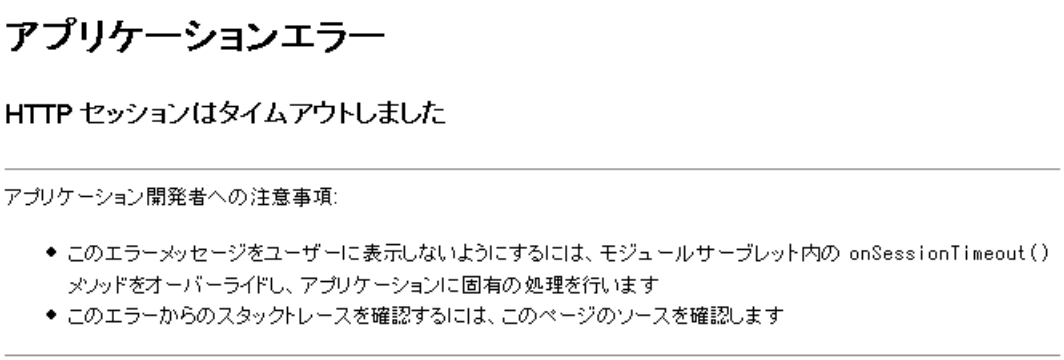
▼ [プロパティ]「固有の URL を生成」ノード

「True」に設定した場合は、設計やテストサイクル中に、ブラウザがキャッシュされたページを使用しないよう、テストの実行のたびにブラウザが起動され、一意の URL が送信されます。これは、テストの実行のたびに一意の ID を持つ名前と値のペアを付加することによって行われます。デフォルト値は「false」で、これは一意の URL を使用しないことを意味します。

▼ [プロパティ]「厳密なセッションタイムアウト」ノード

「True」に設定した場合は、Web アプリケーションフレームワークアプリケーションに強制的にセッションタイムアウト処理コードが実装されます。実装しなかった場合、連続したテスト実行が試みられるたびにブラウザにセッションタイムアウト例外メッセージが表示されます。例外メッセージを回避するには、同じプロセス空間を共有するすべてのブラウザインスタンスを必ず終了させる必要があります (これには、Netscape 同様、メールクライアントが含まれることもある)。

次の図は、ブラウザに表示された HTTP セッションタイムアウトメッセージを示しています。



HTTP セッションタイムアウトの実際のスタックトレースを見るには、メッセージの HTML ソースを表示します。

```
<!-- 例外スタックトレース -->
<!--
javax.servlet.ServletException: This session has timed out
at
com.iplanet.jato.ApplicationServletBase.onSessionTimeout(ApplicationServletBas
e.java:1222)
at
com.iplanet.jato.ApplicationServletBase.fireSessionTimeoutEvent(ApplicationSer
vletBase.java:1079)
at
com.iplanet.jato.ApplicationServletBase.fireSessionEvents(ApplicationServletBa
se.java:834)
at
com.iplanet.jato.ApplicationServletBase.processRequest(ApplicationServletBase.
java:609)
at
com.iplanet.jato.ApplicationServletBase.doPost(ApplicationServletBase.java:474
)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:760)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
...
```

もう 1 つの選択肢は、ログインページにユーザーをリダイレクトするセッションタイムアウト処理ロジックを実装することです。Web アプリケーションフレームワークアプリケーションに自分で作成したセッションタイムアウト処理を実装するには、アプリケーションのアプリケーションサーブレットクラス (このチュートリアル例では `jatotutorial.JatoTutorialAppSerlvletBase`) で `onSessionTimeout` イベントを無効にします。セッションタイムアウト処理の詳細は、『Web アプリケーションフレームワーク 開発ガイド』をご覧ください。

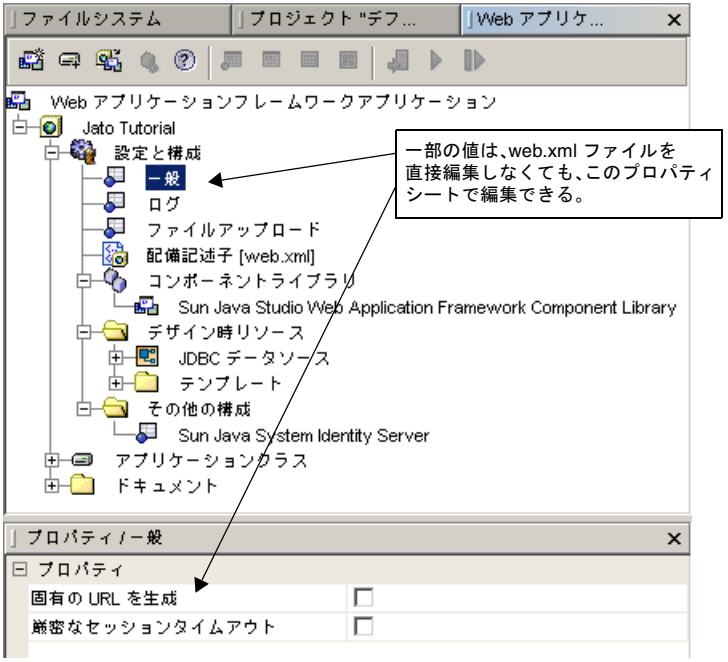
「厳密なセッションタイムアウト」プロパティを「false (デフォルト)」に設定した場合、セッションタイムアウトは無視され、テスト実行は要求があったときに終了します。次の図に示すように、実行時に ModuleServlet クラスに渡されるコンテキストパラメータ (`jato:enforceStrictSessionTimeout`) を配備記述子 (`web.xml`) 内で使用することによって、自動的なセッションタイムアウト処理が実現されます。

```
<web-app>
  <icon>
    <small-icon>sun logo</small-icon>
  </icon>
  <display-name>Jato Tutorial</display-name>
  <context-param>
    <param-name>jato:jatotutorial.main.*:moduleURL</param-name>
    <param-value>../main</param-value>
  </context-param>
  <context-param>
    <param-name>jato:enforceStrictSessionTimeout</param-name>
    <param-value>true</param-value>
  </context-param>
```

生データの web.xml ファイル (配備記述子) で見られる context-param

このパラメータが「false」に設定されるか、存在しない場合、Web アプリケーションフレームワークアプリケーションは、セッションタイムアウト処理として、セッションタイムアウトを無視し、生成ページの最下部に、セッションタイムアウトを明示的に処理するよう開発者に通知するメッセージを表示します。

次の図は、「厳密なセッションタイムアウト」プロパティを設定しているところを示しています。



「ファイルアップロード」ノード

「ファイルアップロード」ノードには、アプリケーションがファイルをアップロードすることを可能にするプロパティがいくつかあります。この機能を使用するには、Servlet v2.3 準拠のサーブレットコンテナでアプリケーションを実行する必要があります。

「ファイルアップロード」ノードのコンテキストメニューコマンド

▼ [なし]

「ファイルアップロード」ノードのプロパティ

詳細は、`com.iplanet.jato.MutipartFormServletFilter` の javadoc を参照してください。

▼ [プロパティ]「一時ファイルを自動削除」

「true」に設定した場合、フィルタは、VM が終了したときの削除対象であることを示すマークを一時ファイルに付けます。この設定はフォールバック戦略として使用することができますが、コンテナの VM が正常に終了して、一時ファイルを削除する保証はありません。また、数百、数千のファイルを作成して削除対象のマークを付けると、確実にコンテナのメモリー負荷が大きくなります。このため、一時ファイルは、アプリケーションがそれらのファイルを必要としなくなったときに、アプリケーションに削除させることを推奨します。コンテナからファイルを削除するには、そのセキュリティポリシーでの「ファイル削除」権限の付与が必要になる場合もあることに注意してください。

▼ [プロパティ]「ファイルアップロードサーブレットフィルタを有効化」

「True」に設定した場合は、編集不可の残りのプロパティのすべてが編集可能になります。

▼ [プロパティ]「ファイルサイズ上限」

アップロードされたファイルの内容の最大バイトサイズです。たとえばユーザーが 1 回の要求で 3 つのファイルをアップロードした場合、どのファイルも、この上限を超えていてはいけません。ただし、これは弱い制限値であることに注意してください。アップロードされた内容がこの上限を超えても、内容が無視されるだけで、要求は引き続き通常の処理が行われ、アプリケーションによるサイズ違反処理を行うことができます。デフォルト値は 1M バイトですが、アプリケーションには、できる限り小さい値に設定することを推奨します。

▼ [プロパティ]「ファイルサイズ上限 (厳格)」

アップロードされたファイルの内容の最大バイトサイズで、このサイズを超えるとエラーが発生し、通常の要求処理は停止します。たとえばユーザーが 1 回の要求で 3 つのファイルをアップロードして、そのうちの少なくとも 1 つのファイルがこの上限を超えていると、すべてのファイルが廃棄され、フィルタはエラー状態の発生を通知します。この厳格な上限値に違反があると、フィルタはエラーハンドラ URL に HTTP リダイレクトを発行します。

リダイレクト URL は、「要求失敗リダイレクト URL」プロパティで設定できます。リダイレクト URL が指定されていない場合、フィルタは例外をスローして、ただちに要求を終了します。デフォルト値は 2M バイトですが、アプリケーションには、できる限り小さい値に設定することを推奨します。

▼ [プロパティ]「要求失敗リダイレクト URL」

ファイルアップロードのサイズ上限を超えるか、要求の構文解析に失敗した場合のクライアントのリダイレクト先の URL です。このパラメータは、現在のアプリケーションの内部または外部の任意の URL でかまいません。指定された URL は、HTTP 302 リダイレクトとして送信されます。

▼ [プロパティ]「要求サイズ上限」

受信要求全体の最大バイトサイズです。要求がこの上限に違反している場合、要求処理は停止し、入力ストリームが廃棄されます。フィルタは、ファイル内容のサイズの厳格な上限値違反のときと同様にこの違反を処理します。デフォルト値は 4M バイトですが、アプリケーションには、できる限り小さい値に設定することを推奨します。

▼ [プロパティ]「一時ファイルディレクトリ」

一時ファイルの作成先のディレクトリです。このパラメータが指定されていない場合は、JDK のデフォルトの一時ディレクトリが使用されます。

▼ [プロパティ]「一時ファイルを使用」

「True」に設定した場合、フィルタは、アップロードされたファイル内容を一時ディレクトリ内の一時ファイルに保存します。「True」以外の場合、アップロードされたファイルの内容は、メモリー上のマルチパートコンテンツオブジェクトにのみ保持されます。

▼ [プロパティ]「デフォルトの文字エンコーディング」

定数 `CHARACTER_ENCODING_OVERRIDE_ATTRIBUTE_NAME` で名前を設定された ServletRequest スコープ属性に値が設定されておらず、サポートされている文字エンコーディングの文字列値が割り当てられている場合のみ、オプションの文字エンコーディングがデフォルトで使用されます。

▼ [プロパティ]「要求された文字エンコーディングを使用」

「True」に設定した場合、定数 `CHARACTER_ENCODING_OVERRIDE_ATTRIBUTE_NAME` で名前を設定された ServletRequest スコープ属性に値が設定されていない場合のみ、ServletRequest.getCharacterEncoding() の値がデコーディングに使用されます。

注意 – ファイルをメモリーに読み込むと、本稼働のスケーラビリティ面で重大な問題が発生する可能性があるため、一時ファイルを使用することを推奨します。デフォルトは「True」です

「ログ」ノード

「ログ」ノードは、Web アプリケーションフレームワークアプリケーションで、使いやすい方法で非常に基本的なトレースを行うことを可能にします。この機能は、com.iplanet.jato.util.Log クラスの動作を利用します。Web アプリケーションフレームワークは、配備記述子ファイル内のコンテキストパラメータを使用して、ログ機能を自動的に有効または無効にします。Web アプリケーションフレームワークは、トレースおよびデバッグ用のログ文をいくつか内蔵していますが、適切なログレベルを持つログ機能のアプリケーションのコードへの追加は開発者が行います。

「ログ」ノードのコンテキストメニューコマンド

▼ [なし]

「ログ」ノードのプロパティ

「ログ」ノードのプロパティはどれも、配備記述子ファイル (Web アプリケーションの WEB-INF ディレクトリ内の web.xml ファイル) に影響します。このため、アプリケーションを再コンパイルしなくても、ログ動作を変更することができます。

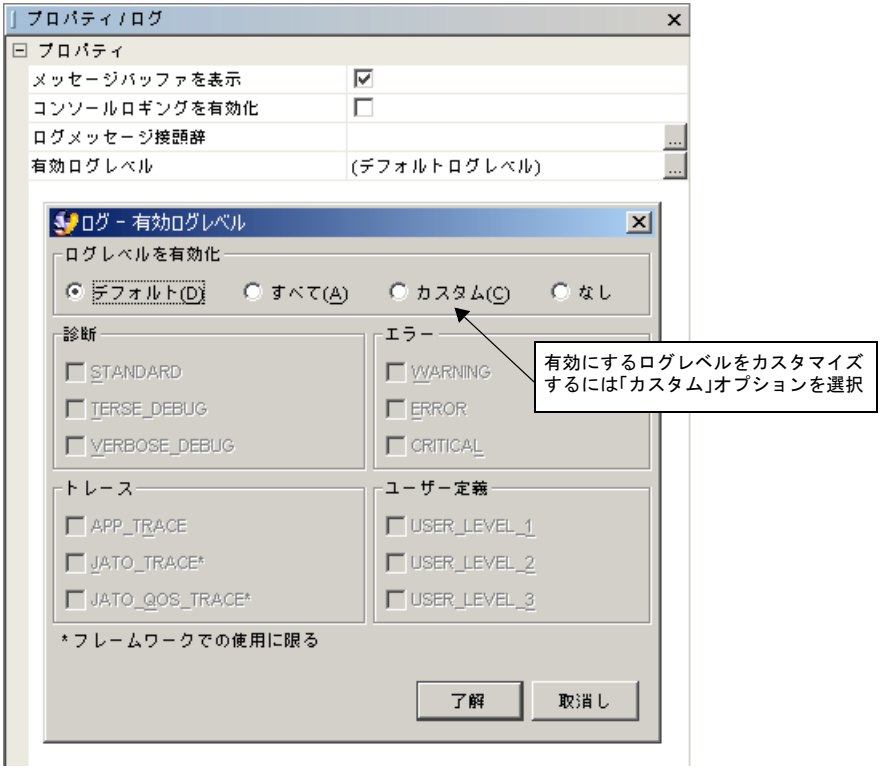
▼ [プロパティ]「コンソールロギングを有効化」

このプロパティは、com.iplanet.JATO.Log クラスの出力を、標準サーブレットコンテキストログ (Sun Java Studio Enterprise 7 の「出力」ウィンドウ、またはサーバー用のコンソールウィンドウ) の他に、サーバーコンソールに送信するかどうかを制御します。

▼ [プロパティ]「有効ログレベル」

このプロパティによって、全ログレベルのうちのデフォルトのサブセット (「デフォルト」) か、すべてのレベル (「すべて」)、あるいはカスタマイズしたサブセット (「カスタム」) を有効にすることができます。また、どのログレベルも有効にしない設定 (「なし」) にすることもできます。

次の図は、「有効ログレベル」プロパティエディタを示しています。



「有効ログレベル」エディタは、「有効ログレベル」プロパティシート値ボックスの省略符号 (...) ボタンをクリックすることによって表示することができます。

```
2003-08-07 14:07:19 [JATO_TRACE] Servlet[jatotutorial_main@3098834]: Enabled
log levels: MANDATORY | STANDARD | VERBOSE_DEBUG | TERSE_DEBUG | JATO_TRACE |
JATO_QOS_TRACE | APP_TRACE | WARNING | ERROR | CRITICAL | USER_LEVEL_1 |
USER_LEVEL_2 | USER_LEVEL_3

2003-08-07 14:07:19 [JATO_TRACE] Servlet[jatotutorial_main@3098834]: Setting
parameter "jato:echoLogToSystemOut" = "true" (java.lang.Boolean)

2003-08-07 14:07:19 [JATO_TRACE] Servlet[jatotutorial_main@3098834]: Setting
parameter "jato:jatotutorial.main.*:moduleURL" = "../main" (java.lang.String)

2003-08-07 14:07:19 [WARNING] Servlet[jatotutorial_main@3098834]: The servlet
"jatotutorial_main" is NOT enforcing strict session timeouts. Be sure to turn
on strict session timeout handling or implement the onSessionTimeout() event
before putting this application into production.

2003-08-07 14:07:19 [JATO_TRACE] Excluded method
"jatotutorial.main.LoginViewBean.beginChildDisplay" from registration

2003-08-07 14:07:19 [JATO_TRACE] Excluded method
"jatotutorial.main.LoginViewBean.endChildDisplay" from registration
```

▼ [プロパティ]「ログメッセージ接頭辞」

フレームワークに組み込まれたログ API を使ってトレースされているログメッセージであることを識別する、カスタマイズ可能な文字列です。この接頭辞文字列は、トレース文の先頭に付加されます。たとえば接頭辞として「*>*>*>」という文字列が入力された場合、トレース文は以下のように表示されます。

```
2003-08-07 14:07:19 *>*>*> [JATO_TRACE] Excluded method
"jatotutorial.main.LoginViewBean.beginChildDisplay" from registration
```

▼ [プロパティ]「メッセージバッファを表示」

appMessage メソッドを使って記録されたメッセージは、「メッセージバッファ」に書き込まれます。メッセージバッファの内容は、表示中の HTML ページの末尾に書き出されます。

以下は、LoginViewBean クラスの beginComponentDisplay イベントで appMessage メソッドを使い、メッセージバッファにメッセージを書き込む例です。

```
public void beginComponentDisplay(DisplayEvent event)
    throws ModelControlException
{
    appMessage("this message is being logged to the Message Buffer");
}
```

appMessage メソッドは、アプリケーションを素早く障害追跡するのに役立ちます。このメソッドは ViewBean や TiledView クラスで非常によく利用され、あらゆるコンテナビュークラスのインスタンスメソッドです。上記の例は、このログメソッドを実装可能な多数ある場所のうちの 1 つにしかすぎません。

次の図は、このメッセージバッファへのログ記録結果を示しています。

「メッセージバッファを表示」を有効にすると、メッセージバッファの内容が表示されます。有効でない場合、バッファは無視されます。つまり、このプロパティは、開発環境ではメッセージを有効にし、本稼働環境ではメッセージを無効にする簡単な手段になります。

「配備記述子」ノード

「配備記述子」ノードは、アプリケーションの WEB-INF ディレクトリにある web.xml ファイルへの単なるリンクです。このファイルは、アプリケーションが Web アプリケーションであることを示します。配備記述子ファイルには、サーブレットコンテナが実行時に Web アプリケーション (Web アプリケーションフレームワーク Web アプリケーションだけではない) を管理するのに使用するさまざまな設定が含まれています。すなわち、表示名やコンテキストパラメータ、サーブレットマッピング、taglib 宣言など多くの情報が含まれています。Web アプリケーションフレームワークのツールを利用することによって、Web アプリケーションの開発初心者は、このファイルの細々したことを無視しながら正しく作業を進めることができま

す。ただし、J2EE に精通している開発者は、このファイルをテキストまたは XML エディタで開いて直接編集したり、あるいは Sun Java Studio Enterprise 7 でこのノードを選択してプロパティシートを使用して編集することもできます。

「配備記述子」ノードのコンテキストメニューコマンド

▼ [アクション]「編集」

このアクションは、Sun Java Studio Enterprise 7 のエディタで配備記述子を開きます。このファイルを変更するときは、十分に注意してください。エントリの一部は、Web アプリケーションフレームワークのツールで管理されます。変更したエントリに誤りがあると、ツールがそのエントリを正しく認識できなくなることがあります。

「配備記述子」ノードのプロパティ

「配備記述子」ノードのプロパティはすべて、配備記述子の要素エントリに 1 対 1 で対応しています。Web アプリケーションフレームワークのツールモジュールは、Sun Java Studio Enterprise 7 の Web モジュールがすでに提供している以外のプロパティをいっさい追加しません。「配備記述子ファイル」ノードのプロパティに関する詳細は、Web モジュールのマニュアルを参照してください。

「コンポーネントライブラリ」ノード

「コンポーネントライブラリ」ノードは、Web アプリケーションフレームワークアプリケーションの `WEB-INF/lib` ディレクトリにマウントされている Web アプリケーションフレームワークの全コンポーネントライブラリの JAR ファイルの一覧です。あらゆる Web アプリケーションフレームワークアプリケーションに、Web アプリケーションフレームワーク Standard Component Library (WAF SCL) が含まれます。

開発者は再利用可能な独自の Web アプリケーションフレームワークコンポーネントライブラリを作成したり、Sun 以外のコンポーネントベンダーからそうしたライブラリを購入したりできるようになるでしょう。Web アプリケーションフレームワークアプリケーションの `WEB-INF/lib` ディレクトリにコンポーネントライブラリの JAR ファイルが書き込まれると、Sun Java Studio Enterprise 7 がその新しいライブラリを認識してマウントし、このノードの下に表示されます。

コンポーネントライブラリのこの一覧は、Sun Java Studio の Web アプリケーションフレームワークアプリケーションで利用可能なコンポーネントライブラリを素早く参照するための手段であり、どのような方法でも変更することはできません。Web アプリケーションフレームワークコンポーネントの作成についての詳細は、『Web アプリケーションフレームワーク コンポーネント作成ガイド』をご覧ください。

「コンポーネントライブラリ」ノードのコンテキストメニューコマンド

▼ [アクション]「コンポーネントライブラリを追加」

Web アプリケーションフレームワークアプリケーションのルートノードの同じ名前のアクションと同じです (上記の [アクション]「コンポーネントライブラリを追加」を参照)。

「コンポーネントライブラリ」ノードのプロパティ

▼ [なし]

「コンポーネントライブラリ」ノードのサブノード

「コンポーネントライブラリ」ノードのサブノードは、Web アプリケーションフレームワーク Standard Component Library (Web アプリケーションフレームワークに付属) か、入手して Web アプリケーションフレームワークアプリケーションに追加した Sun 以外の Web アプリケーションフレームワークコンポーネントライブラリ、あるいは作成して Web アプリケーションフレームワークアプリケーションに追加したカスタムの Web アプリケーションフレームワークコンポーネントライブラリのいずれかです。Web アプリケーションフレームワークコンポーネントの作成についての詳細は、『Web アプリケーションフレームワーク コンポーネント作成ガイド』をご覧ください。

「デザイン時リソース」ノード

このノードには、「JDBC データソース」ノードと「テンプレート」ノードの 2 つのサブノードがあります。

「デザイン時リソース」ノードのコンテキストメニューコマンド

新しい JDBC データソースは、「JDBC データソース」ノードを右クリックし、「JDBC データリソース追加」を選択することによって作成することができます。作成したデータソースは、削除することはできますが、変更することはできません。

「デザイン時リソース」ノードのプロパティ

▼ [なし]

「デザイン時リソース」ノードのサブノード

「デザイン時リソース」ノードには、「JDBC データソース」ノードと「テンプレート」ノードの 2 つのサブノードがあります。

「JDBC データソース」ノード

「JDBC データソース」ノードは、JDBC データソースウィザードを使い、Web アプリケーションフレームワークアプリケーションに作成された全 JDBC データソースの一覧です。これらのデータソースは、ウィザードを使って JDBC SQL 表とストアドプロシージャモデルを構築する際、表と列、ストアドプロシージャを選択できるよう、ターゲットデータベースのスキーマデータに収集するために設計時にのみ使用されます。これらデータソースはデータベース接続がどのように作成されるかには関係なく、実行時にサーブレットコンテナや Web アプリケーションフレームワークアプリケーションによって取得されることはありません。ターゲットの本稼働サーブレットコンテナに用意されているツールを使用して、適切な実行時 JDBC および JNDI 設定を行う必要があります。

「JDBC データソース」ノードのコンテキストメニューコマンド

▼ [アクション]「JDBC データリソース追加」

このアクションは、「実行時」ウィンドウの「データベース」ノード下に設定されているデータベース接続に基づいて、新しい Web アプリケーションフレームワークデータソースを作成します。新しい Web アプリケーションフレームワークデータソースを作成するにあたっては、事前にデータベース接続が作成されていて、データベースサーバーが動作し、アクセス可能であることを確認してください。

「JDBC データソース」ノードのプロパティ

▼ [なし]

「JDBC データソース」ノードのサブノード

「JDBC データソース」ノードのサブノードになるのは、実際に作成された Web アプリケーションフレームワークデータソースだけです。

Web アプリケーションフレームワークデータソースノードのコンテキストメニューコマンド

▼ [アクション]「削除」

作成した Web アプリケーションフレームワークデータソースに対して行える操作は、削除だけです。Web アプリケーションフレームワークデータソースを変更する必要がある場合は、いったん削除して、再作成する必要があります。

Web アプリケーションフレームワークデータソースノードのプロパティ

▼ [なし]

「テンプレート」ノード

「テンプレート」ノードは、新しい JSP ページ/ページレットファイルを生成する際にページ/ページレットコンポーネントウィザードが使用する全テンプレートの格納場所です。このテンプレートの一覧にカスタム JSP パージ/ページレットを追加することができます。

「テンプレート」ノードのコンテキストメニューコマンド

「テンプレート」ノードは、一般的な Java パッケージノードとして表示されます。このノードには、他のフォルダノードに用意されているすべてのフォルダアクションが用意されています。

「テンプレート」ノードのプロパティ

「テンプレート」ノードは、一般的な Java パッケージノードとして表示されます。このノードには、他のフォルダノードに用意されているすべてのフォルダプロパティが用意されています。

「テンプレート」ノードのサブノード

「テンプレート」ノードでは、追加されたレベルのフォルダと、実際の JSP ページ/ページレットテンプレートをそのサブノードとすることができます。

「Sun Java System Identity Server」ノード

「Sun Java System Identity Server (旧称「Sun ONE Identity Server」) ノードには、アプリケーションが Identity Server のセキュリティ機能を使用するよう設定することを可能にするプロパティがいくつかあります。これらのプロパティは、Sun Java System Application Server (旧称「Sun ONE Application Server」) を使用してアプリケーションを配備した場合にのみ適用できます。

「Sun Java System Identity Server」ノードのコンテキストメニューコマンド

▼ [なし]

「Sun Java System Identity Server」ノードのプロパティ

「セキュリティ制限」、「セキュリティロール」、および「マップされたセキュリティロール」プロパティは、配備記述子ファイル (Web アプリケーションの WEB-INF ディレクトリ内の web.xml) のノードに表示されるプロパティと同じものです。便宜を考えて、これらのプロパティは「Sun Java System Identity Server」ノードにも表示されます。

▼ [プロパティ]「Identity Server のセキュリティを有効化」

このプロパティは、Web モジュールが Identity Server のセキュリティ機能を使用できるようにするかどうかを指定します (「True」/「False」) デフォルト値は「False」です。

▼ [プロパティ]「セキュリティ制限」

「セキュリティ制限」プロパティは、Web モジュールに定義されているセキュリティ制限数を示します。セキュリティ制限とは、Web リソースのロールグループまたはトランスポート要件へのマッピングです。デフォルト値は「セキュリティ制限なし」です。セキュリティ制限を指定するには、「セキュリティ制限」プロパティ値フィールドをクリックし、省略記号ボタンをクリックして、「セキュリティ制限」プロパティエディタを表示します。

▼ [プロパティ]「セキュリティロール」

「セキュリティロール」プロパティは、Web モジュールに定義されているセキュリティロール数を示します。ロールは、アプリケーション開発者またはアプリケーション構築者が定義した抽象的で論理的なユーザーグループ分けです。セキュリティロールは、Web モジュール内のさまざまなリソースへのアクセスを許可したり、拒否したりできます。デフォルト値は「セキュリティロールなし」です。セキュリティロールを指定するには、「セキュリティロール」プロパティ値フィールドをクリックし、省略記号ボタンをクリックして、「セキュリティロール」プロパティエディタを表示します。

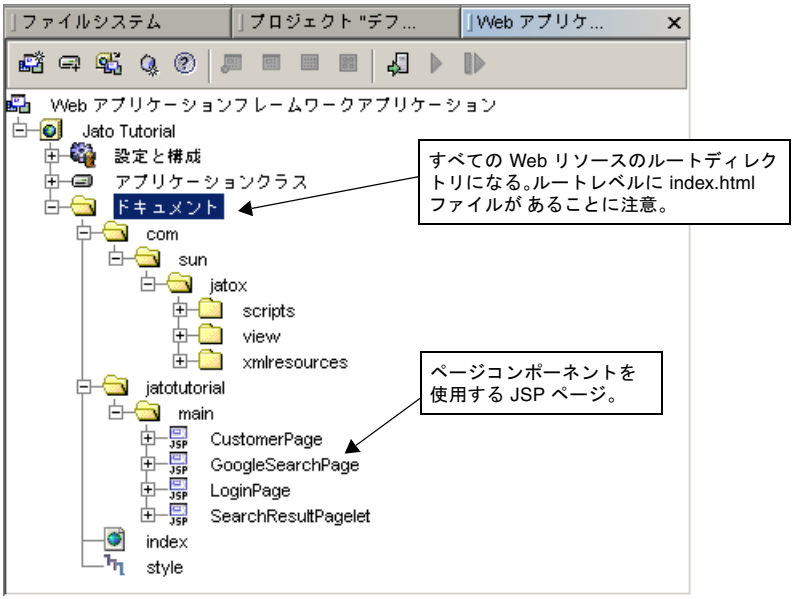
▼ [プロパティ]「マップされたセキュリティロール」

「マップされたセキュリティロール」プロパティは、Web モジュールの現在アクティブなレルムにおいて主体 (ユーザー) またはグループにマップされているロールを示します。このプロパティは、Sun Java System Application Server グループか主体、またはその両方に論理的な J2EE ロールをマップします。デフォルト値は「ロールなし」です。主体またはグループにロールをマップするには、「マップされたセキュリティロール」値フィールドをクリックし、省略記号ボタンをクリックして、「マップされたセキュリティロール」エディタを表示します。

「ドキュメント」ノード

「ドキュメント」ノードは、HTML や JSP、CSS、イメージなどの Web リソースを含む Web ドキュメントのルートです。

次の図は、すべての Web リソースのルートディレクトリである、「ドキュメント」ノードを示しています。



Web アプリケーションフレームワークの JSP ページが、アプリケーション内のページおよびページレットコンポーネントのパッケージと相似のディレクトリレイアウトで構造化されていることに注意してください。この構造にすることを推奨しますが、実際には JSP は、サーブレットコンテキストディレクトリの構造内の任意の場所に配置することができます。

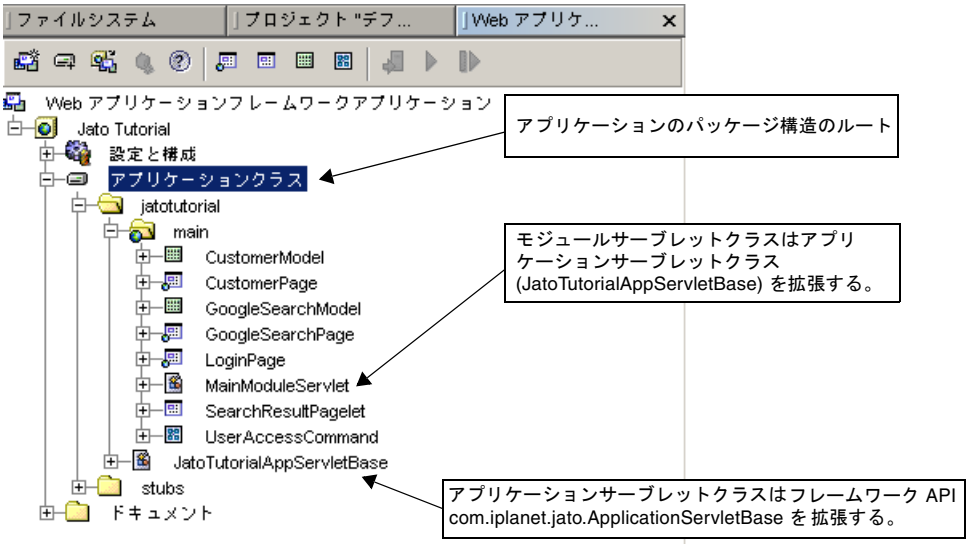
相似のディレクトリ構造にする主な理由は 2 つあります。第 1 に、開発者、特に Web アプリケーションフレームワークの初心者が ViewBean を使用する JSP を見つけるのが容易になります。どのページもクラスと JSP の両方で構成されているため、クラスと JSP のディレクトリ構造は同じにした方が簡単です。一方の場所が分かっていれば、ただちにもう一方の正確な場所も分かります。第 2 に、これは最も重要なことで、この相似のディレクトリ構造は、当然アプリケーションコードと同じ名前空間スコープ規則を継承します。このため、アプリケーションに複数のモジュールが存在する場合に、同じ名前を持つクラスと JSP が衝突することがなくなります。つまり、完全にモジュールの独立性が得られます。

この実例として、モジュール 1 に「menu」というページ (`MenuViewBean.java` と Menu.jsp) があり、もう 1 つのモジュール 2 にも menu ページがあると仮定しましょう。相似ディレクトリ構造でない場合、2 つの menu JSP ファイルは、たとえば `Menu.jsp` と `Menu2.jsp` というように一意のファイル名である必要があります。

「アプリケーションクラス」ノード

「アプリケーションクラス」ノードには、Web アプリケーションフレームワークアプリケーションの Java パッケージ構造がまるごと含まれます。「アプリケーションクラス」ノードは、アプリケーションのルートパッケージです。

次の図は「アプリケーションクラス」ノードを示しています 。



「アプリケーションクラス」ノードのコンテキストメニューコマンド

このノードで必要になる可能性があるアクションは、「コンパイル」または「構築」アクションだけです。

▼ [アクション]「コンパイル」

このアクションは、現在のディレクトリ内の新しいファイルまたは前回コンパイル後に変更されたファイルをコンパイルします。最新かどうかは、コンパイルのソース (`.java`) と生成ファイル (`.class`) のタイムスタンプを比較することによって調べます。サブフォルダ内のファイルはコンパイルされません。

▼ [アクション]「すべてをコンパイル」

このアクションは、サブフォルダ内のファイルを含めて、新しいファイルまたは前回コンパイル後に変更されたファイルをコンパイルします。

▼ [アクション]「構築」

このアクションはフォルダ内の .class ファイルを削除し、ソースファイルを再コンパイルします。サブフォルダ内の .class ファイルを削除することはなく、ソースファイルのコンパイルも行われません。

▼ [アクション]「すべてを構築」

このアクションは、フォルダとそのサブフォルダ内のすべての .class ファイルを削除し、すべてのファイルをコンパイルします。

「アプリケーションクラス」ノードのプロパティ

▼ [プロパティ]「モジュールパッケージのみ表示」

このプロパティは、モジュール以外のすべてのパッケージ (配備記述子でモジュールサーブレットとして宣言されているサーブレットが存在しないパッケージ) を隠すことを可能にします。これは、パッケージ構造が深くて複雑で、モジュールのフォルダのマークが付けられたパッケージのみ見たい場合に便利なことがあります。

「アプリケーションクラス」ノードのサブノード

「アプリケーションクラス」ノードのサブノードになるのは、標準的な Java パッケージのフォルダと Web アプリケーションフレームワークモジュールのフォルダ (Java パッケージでもある)、および Java クラスです。

Java パッケージフォルダのノード

「アプリケーションクラス」ノードの下のフォルダはすべて Java パッケージのフォルダです。どのように複雑なパッケージ構造のアプリケーションでも設計、開発することができます。そうしたパッケージの少なくとも 1 つを、Web アプリケーションフレームワークモジュールのフォルダとして宣言します。モジュールのフォルダについては、この後詳しく説明します (モジュールフォルダのノードを参照)。

Java パッケージフォルダノードのコンテキストメニューコマンド

▼ [アクション]「コンパイル」

このアクションは、現在のディレクトリ内の新しいファイルまたは前回コンパイル後に変更されたファイルをコンパイルします。最新かどうかは、コンパイルのソース (.java) と生成ファイル (.class) のタイムスタンプを比較することによって調べます。サブフォルダ内のファイルはコンパイルされません。

▼ [アクション]「すべてをコンパイル」

このアクションは、サブフォルダ内のファイルを含めて、新しいファイルまたは前回コンパイル後に変更されたファイルをコンパイルします。

▼ [アクション]「構築」

このアクションはフォルダ内の .class ファイルを削除し、ソースファイルを再コンパイルします。サブフォルダ内の .class ファイルを削除することはなく、ソースファイルのコンパイルも行われません。

▼ [アクション]「すべてを構築」

このアクションは、フォルダとそのサブフォルダ内のすべての .class ファイルを削除し、すべてのファイルをコンパイルします。

▼ [アクション]「追加」

このアクションは、他のメニューアクション (「コマンド」、「モデル」、「モジュールフォルダ」、「ページ (ViewBean)」、「サブページ (ContainerView)」) を選択するためのものです。これらのオプションはすべて、それぞれのコンポーネントを作成する際の案内役をするウィザードを起動します。これらのアクションは、Web アプリケーションフレームワークのツールバーのボタン (「Web アプリケーションフレームワークアプリケーション」ウィンドウの最上部) を使用したり、Sun Java Studio Enterprise 7 のメニューオプションの「ファイル」-「新規」を選択して「Web アプリケーションフレームワーク」ノードを開き、コンポーネント作成ウィザードのいずれかを選択した場合と同様の働きをします。

「アプリケーションクラス」ノード下のフォルダノードを右クリックすると、別のアクションもいくつか用意されています。「追加」メニュー項目は、一部 Web アプリケーションフレームワークコンポーネントウィザードに対するショートカットです。

一部の Web アプリケーションフレームワークコンポーネントは、モジュールフォルダ内で作成する必要があります (「モジュールフォルダのノード」を参照)。ウィザードは、この条件を持つコンポーネントが作成されるのを防ぎます。

▼ [アクション]「モジュールに変換」

このアクションは、通常の Java パッケージを Web アプリケーションフレームワークモジュールフォルダに変換します。このアクションを選択すると、標準パッケージをモジュールフォルダに変換するための入力用のダイアログが表示されます。ターゲットパッケージにサーブレットクラスが存在する場合、それらクラスは「モジュールサーブレット候補」として表示され、サーブレットクラスが存在しない場合は、クラス名を入力して新しいモジュールサーブレットクラスを作成することができます。

モジュールフォルダの新規作成は、作成先のベースパッケージのノード (「アプリケーションクラス」か「jatotutorial」、「main」あるいは「アプリケーションクラス」内の他のパッケージ) を右クリックして、「追加」を選択し、「モジュールフォルダ」を選択することによって行うことができます。これで、新しいモジュールサーブレット名の入力が求められます。

Java パッケージフォルダのプロパティ

▼ [プロパティ]「モジュール」

Web アプリケーションフレームワークアプリケーション内のモジュール以外の Java パッケージは、このプロパティを「False」から「True」に変更することによってモジュールフォルダに変換することができます。詳細は、[アクション]「モジュールに変換」を参照してください。

▼ [プロパティ]「名前」

フォルダ名です。

▼ [プロパティ]「ソートモード」

モジュールのサブノード、あるいはエクスプローラ内のあらゆるフォルダはソート解除したり、種類や名前、名前 (パッケージ名が先) でソートしたりできます。

モジュールフォルダのノード

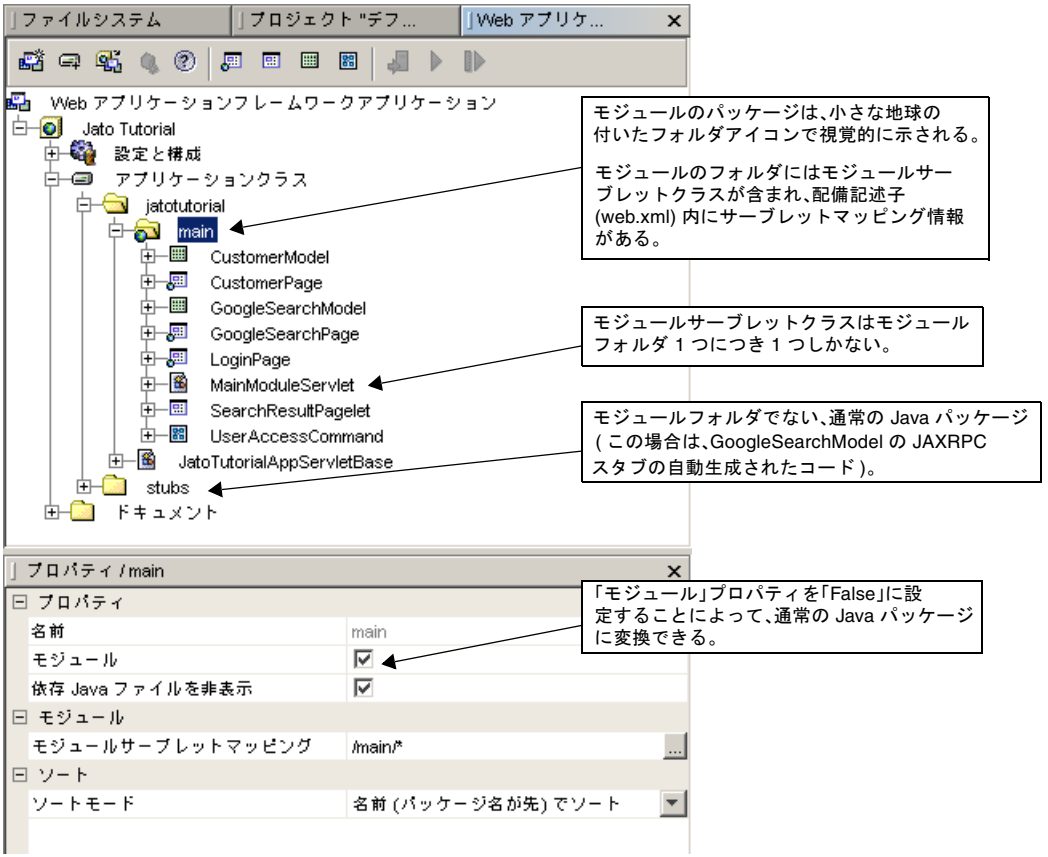
Web アプリケーションフレームワークモジュールフォルダのノードは、従来形式の Java パッケージフォルダの左下隅に緑の小さな地球の印の付いたノードです。Java パッケージフォルダに Web アプリケーションフレームワークモジュールサーブレットが含まれていて、そのモジュールサーブレットが配備記述子ファイル (Web アプリケーションの `WEB-INF` ディレクトリ内にある (`web.xml` ファイル) に正しく登録されている場合、IDE はそのパッケージフォルダを正当な Web アプリケーションフレームワークモジュールフォルダと認識します。そして、正当な Web アプリケー

ションフレームワークモジュールフォルダの作成を自動的に管理します。IDE はまた、Java パッケージフォルダの Web アプリケーションフレームワークモジュールフォルダへの自動変換もサポートしています。

一般に、Web アプリケーションフレームワークアプリケーションのノードには、少なくとも 1 つの Web アプリケーションフレームワークモジュールフォルダが含まれます。大規模な Web アプリケーションフレームワークアプリケーションになると、複数の Web アプリケーションフレームワークモジュールフォルダが含まれることがあります。モジュールフォルダは、大規模なアプリケーションを論理的に分割することを可能にします。

一般にモジュールフォルダには、ページやページレット、モデルなどの Web アプリケーションフレームワークコンポーネントが含まれます。また、あらゆる Java パッケージフォルダに入れることが可能な任意のアプリケーションリソースも含まれます。本質的には、Web アプリケーションフレームワークモジュールフォルダの働きは、Web アプリケーションフレームワークページコンポーネントが Web アプリケーションフレームワークモジュールフォルダ内にあるときにのみ、それらコンポーネントを実行可能にすることにあります。Web アプリケーションフレームワークのページやページレット、モデル、コマンドコンポーネントは、任意の Java パッケージフォルダに作成することができますが、Web アプリケーションフレームワークページコンポーネントは、Web アプリケーションフレームワークフォルダの中からしか実行できません。

次の図はモジュールフォルダのノードを示しています。



モジュールフォルダノードのコンテキストメニューコマンド

▼ [アクション]「コンパイル」

このアクションは、現在のディレクトリ内の新しいファイルまたは前回コンパイル後に変更されたファイルをコンパイルします。最新かどうかは、コンパイルのソース (.java) と生成ファイル (.class) のタイムスタンプを比較することによって調べます。サブフォルダ内のファイルはコンパイルされません。

▼ [アクション]「すべてをコンパイル」

このアクションは、サブフォルダ内のファイルを含めて、新しいファイルまたは前回コンパイル後に変更されたファイルをコンパイルします。

▼ [アクション]「構築」

このアクションはフォルダ内の .class ファイルを削除し、ソースファイルを再コンパイルします。サブフォルダ内の .class ファイルを削除することはなく、ソースファイルのコンパイルも行われません。

▼ [アクション]「すべてを構築」

このアクションは、フォルダとそのサブフォルダ内のすべての .class ファイルを削除し、すべてのファイルをコンパイルします。

▼ [アクション]「追加」

このアクションは、他のメニューアクション (「コマンド」、「モデル」、「モジュールフォルダ」、「ページ (ViewBean)」、「サブページ (ContainerView)」) を選択するためのものです。これらのオプションはすべて、それぞれのコンポーネントを作成する際の案内役をするウィザードを起動します。これらのアクションは、Web アプリケーションフレームワークのツールバーのボタン (「Web アプリケーションフレームワークアプリケーション」ウィンドウの最上部) を使用したり、Sun Java Studio Enterprise 7 のメニューオプションの「ファイル」-「新規」を選択して「Web アプリケーションフレームワーク」ノードを開き、コンポーネント作成ウィザードのいずれかを選択した場合と同様の働きをします。

「アプリケーションクラス」ノード下のフォルダノードを右クリックすると、別のアクションもいくつか用意されています。「追加」メニュー項目は、一部 Web アプリケーションフレームワークコンポーネントウィザードに対するショートカットです。

モジュールフォルダノードのプロパティ

▼ [プロパティ]「モジュール」

モジュールのフォルダには、「モジュール」というプロパティがあります。モジュールの場合、このプロパティは必ず「True」です。このことは、このフォルダに ModuleServlet クラス (この場合は MainModuleServlet) があり、配備記述子 (web.xml) 内にそのエントリがあることを意味します。

このエントリの内容は以下のようになっています。

```
<context-param>
     <param-name>jato:jatotutorial.main.*:moduleURL</param-name>
     <param-value>../main</param-value>
 </context-param>

 <servlet>
     <servlet-name>jatotutorial_main</servlet-name>
     <servlet-class>jatotutorial.main.MainModuleServlet</servlet-class>
     </servlet>
 <servlet-mapping>

 <servlet-name>jatotutorial_main</servlet-name>
     <url-pattern>/main/*</url-pattern>
 </servlet-mapping>
```

「モジュール」プロパティを「False」に変更すると、配備記述子からこうしたエントリが削除され、モジュールフォルダは通常の Java パッケージになります。モジュールフォルダに戻すには、「モジュール」プロパティを「True」に設定し直すか、モジュール以外のフォルダを右クリックして、「モジュールに変換」アクションを選択します。詳細は、[アクション]「モジュールに変換」を参照してください。

注 – Web アプリケーションフレームワークのツールでは、配備記述子ファイルを直接編集することなく、そのエントリの一部をカスタマイズすることができます。ただし、Web アプリケーションの配備記述子や Web アプリケーションフレームワークに精通しているならば、ファイルを直接編集してもかまいません。変更したエントリに誤りがあると、設計時および実行時環境で予期しない結果が生じることがありますので注意してください。

▼ [プロパティ]「モジュールサーブレットマッピング」

このプロパティは、ユーザー要求に含まれる URL パターンのサーブレット (モジュールサーブレット) へのマッピングに基づいてサーブレットを簡単に起動できるようにします。たとえば `/main/*` はそのパターンであると仮定します。このことは、`http://<server>/<servletcontext>/main/<anything-at-all>` というパターンが含まれている URL のどれによっても、`MainModuleServlet` サーブレットクラスが呼び出されることを意味します。

▼ [プロパティ]「ソートモード」

モジュールのサブノード、あるいはエクスプローラ内のあらゆるフォルダはソート解除したり、種類や名前、名前 (パッケージ名が先) でソートしたりできます。

第3章

Web アプリケーションフレームワークコンポーネントのノード

この章では、Web アプリケーションフレームワークアプリケーションで作成する主な Web アプリケーションフレームワークコンポーネントが視覚的に表現するさまざまなノードの概要を説明します。

この章では、ノードの詳細とともに、Web アプリケーションフレームワーク固有のコンテキストメニューのアクションについても詳しく説明します。

アプリケーションに固有のページやページレット、モデル、コマンドコンポーネントはそれぞれ、アプリケーションコンポーネントの作成段階で選択された、対応する Web アプリケーションフレームワークコンポーネントのサブタイプです。IDE は内部的にベースのコンポーネントを調べ、アプリケーションコンポーネントのもとになっているライブラリコンポーネントによって宣言されているコンポーネントプロパティを動的に提供します。この章では、任意の種類のあらゆるコンポーネントに共通するコンテキストメニューコマンドおよびプロパティを重点的に説明します。

ページコンポーネントノード

Web アプリケーションフレームワークのページコンポーネントは、Web アプリケーションフレームワークアプリケーションの可視形式の一次オブジェクトです。Web アプリケーションフレームワークのあらゆるページコンポーネントは、実行時に URL で呼び出すことができ、ドキュメントを返します。

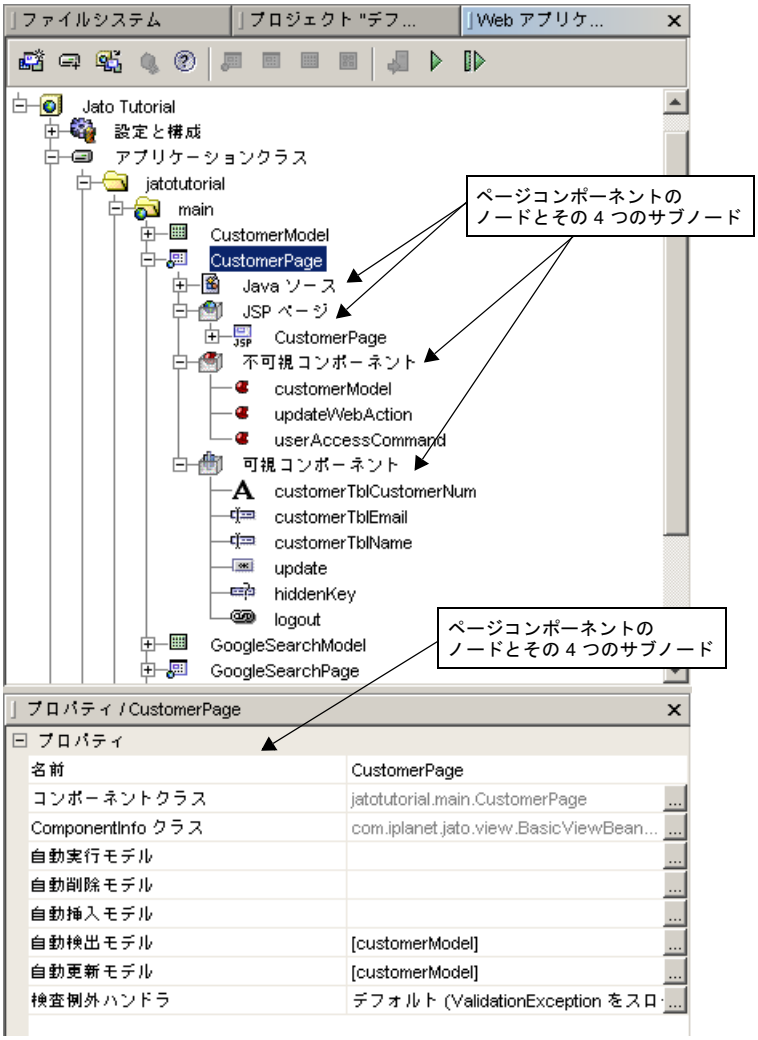
Web アプリケーションフレームワークページコンポーネントは、1 つの Java クラスと関連付けられている JSP から構成されます (JSP が関連付けられている場合)。IDE はサブノードの集まりとしてページの構造を提示することによって Web アプリケーションフレームワークページの作成と構成の手助けをします。以下では、それらサブノードについて説明します。

コンポーネントのサブノードが追加、削除されたり、宣言することによって作成されたりすると、IDE は自動的にそのコンポーネントの Java ソースファイルを更新して、コンポーネントの構成との同期が維持されるようにします。自動的に生成された

Java ソースコードは保護ブロックで守られ、ユーザーが変更できないようになっています。コンポーネントの構成に変化があるたびに再生成されることがあるため、IDE は保護ブロックの編集を許可しません。

アプリケーションの各ページコンポーネントは、ページの作成段階で選択された Web アプリケーションフレームワークページコンポーネントのサブタイプです。IDE はその内部で、ベースのページコンポーネントを調べ、アプリケーションコンポーネントのもとになっているライブラリコンポーネントによって宣言されているコンポーネントプロパティを動的に提供します。

次の図は、ページコンポーネントノードとその 4 つのサブノードの例です。



ページコンポーネントノードのコンテキストメニューコマンド

▼ [アクション]「開く」

このアクションは、ソースエディタでページコンポーネントの Java ソースファイルを開き、そのクラスにジャンプします。

▼ [アクション]「デフォルト JSP を編集」

このアクションはソースエディタでページコンポーネントのデフォルトの JSP ファイルを開いて、そのページにジャンプします。各ページコンポーネントには JSP が関連付けられていることもあれば、関連付けられていないこともあります。複数関連付けられている場合は、そのうちの 1 つがデフォルト JSP として指定されます。

▼ [アクション]「エラー情報」

このアクションは、1 つ以上のエラーメッセージの一覧をダイアログで表示します。これらのエラーメッセージは構成の矛盾点を示すため、これによってコンポーネントの構成またはテスト実行を続ける前に対策を講じることができます。

このコマンドは、コンポーネントの内部状態が構成矛盾によるエラーの場合にのみノードのコンテキストメニューに表示されます。コンポーネントの内部状態がエラーの場合、そのノードには必ず標準のエラーまたは警告の印が付けられます。この警告またはエラーの印は、エラーを訂正すると消えます。

▼ [アクション]「イベント」

このアクションは、このコンポーネントへの実装に適したイベントハンドラメソッドの一覧からなるサブメニューを起動します。すでに実装されているイベントハンドラの名前は太字で表示されます。実装されていないメソッドの名前は太字で表示されません。

一覧から太字のイベントハンドラを選択すると、IDE がコンポーネントの Java ソースを開き、そのイベントハンドラの先頭にカーソルを移動します。太字でないイベントハンドラを選択すると、IDE がコンポーネントの Java ソースファイルに、対応するイベントハンドラのメソッドスタブを追加し、その上にカーソルを移動します。

利用可能なイベントハンドラメソッドの一覧はコンポーネントによって異なり、現在のコンポーネントのもとになっているコンポーネントによって宣言されています。コンポーネントによっては、イベントハンドラを宣言していないものもあります。コンポーネントがイベントハンドラを 1 つも宣言していない場合、このメニューコマンドは使用不可になります。

▼ [アクション]「コンポーネントメソッド」

このアクションは、このコンポーネントが実装可能な、有用なコンポーネントメソッドの一覧からなるサブメニューを起動します。すでに実装されているコンポーネントメソッドの名前は太字で表示されます。実装されていないメソッドの名前は太字で表示されません。

一覧から太字のコンポーネントメソッドを選択すると、IDE がコンポーネントの Java ソースを開き、そのメソッドの先頭にカーソルを移動します。太字でないコンポーネントメソッドを選択すると、IDE がコンポーネントの Java ソースファイルに対応するメソッドスタブを追加し、その上にカーソルを移動します。

利用可能なコンポーネントメソッドの一覧はコンポーネントによって異なり、現在のコンポーネントのもとになっているコンポーネントによって宣言されています。コンポーネントによっては、コンポーネントメソッドを宣言していないものもあります。一般にコンポーネントメソッドは、利用可能なベースクラスメソッドのサブセットです。そのサブセットは、特にサブクラスによるオーバーライドを意図したもので、ライブラリコンポーネントの作成者がこの方法を使用して公開することに意味があるものです。コンポーネントがイベントハンドラを 1 つも宣言していない場合、このメニューコマンドは使用不可になります。

▼ [アクション]「ページを実行」

このアクションはページをテスト実行します。このコマンドはブラウザを起動し、適切な URL を呼び出して IDE に現在設定されているサーブレットコンテナに現在のページを要求します。このコマンドはアプリケーションを配備しません。前回の配備以降に加えられた変更はテスト実行に反映されません。

▼ [アクション]「ページを実行 (再配備)」

このアクションはページをテスト実行します。このコマンドは現在のアプリケーションを自動的にコンパイルして、IDE に現在設定されているサーブレットコンテナに配備してから、ブラウザを起動し、適切な URL を呼び出して現在のページを要求します。

▼ [アクション]「名前を変更」

このアクションはコンポーネントクラス名を設定して、コンポーネントの Java ソースファイルの名前を変更します。

▼ [アクション]「削除」

このアクションはページコンポーネントを削除します。コンポーネントの Java ソースファイルも削除します。ページに JSP ファイルが関連付けられている場合は、その JSP ファイルも削除するかどうか問い合わせます。

▼ [アクション]「カット」、「コピー」、「ペースト」

従来のカット、コピー、ペースト動作をするアクションです。

コピー元のアプリケーション内の他のオブジェクトに対する参照がコピー元のページに含まれている場合、Web アプリケーションフレームワークアプリケーション間のコピーは行わないでください。異なる Web アプリケーションフレームワークアプリケーション間では、参照が維持されません。

ページコンポーネントノードのプロパティ

▼ [プロパティ]「コンポーネントクラス」

このプロパティは、コンポーネントの完全限定クラス名を示します。このプロパティは編集できません。

▼ [プロパティ]「ComponentInfo クラス」

このアクションは、コンポーネントの完全限定 `ComponentInfo` クラスを示します。このプロパティは編集できません。

▼ [プロパティ]「名前」

このプロパティは、コンポーネントクラス名を設定します。

ページコンポーネントのサブノード

ページノードのすぐ下には、以下の 4 つのサブノードがあります。

- 「Java ソース」
- 「JSP ページ」
- 「不可視コンポーネント」
- 「可視コンポーネント」

これらのノードはそれぞれ専用のサブノードを持つことができます。

「Java ソース」ノード

「Java ソース」ノードは、このコンポーネントの Java ソースファイルにノードレベルでアクセスできるようにします。Web アプリケーションフレームワークコンポーネントのノードは 「Java ソース」ノードの親になり、アプリケーションコンポーネ

ントが論理的に単なる Java ノード 以上のものから構成されていることを強調します。以下の点を除けば、「Java ソース」ノードは、Sun Java Studio Enterprise 7 の従来の Java ソースコードのノードと同じです。

- Web アプリケーションフレームワークコンポーネントのノードが親である。
- ノード名は固定で、「Java ソース」である。
- コンテキストメニューの「カット」、「削除」、「名前を変更」コマンドは使用不可になる。

  これらのコマンドについては、コンテキストメニューではなく、コンポーネントのカット、削除、名前を変更コマンドを使用します。

「Java ソース」ノードのコンテキストメニューコマンドとプロパティについては、Sun Java Studio Enterprise 7 のマニュアルに完全な説明があります。

「JSP ページ」ノード

「JSP ページ」ノードは、ページコンポーネント (ページコンポーネントと関連付けられている JSP) を使用しているすべての JSP の一覧です。ページには JSP が関連付けられていることもあれば、JSP に関連付けられないこともあります。このため、このノードには JSP サブノードがないこともあれば、複数含まれることもあります。実行時のクライアント要求があると、関連付けられている 1 つの JSP を使ってページのコンテンツが生成されます。ただし、Web アプリケーションフレームワークのページは、要求に対して関連付けられている JSP のうちの任意のものを条件付きで動的に利用するようコーディングすることができます。この Web アプリケーションフレームワーク機能は並列コンテンツ (parallel content) といいます。並列コンテンツの詳細は、『Web アプリケーションフレームワーク 開発ガイド』をご覧ください。

JSP が Web アプリケーションフレームワークページに関連付けられると、IDE はそのページコンポーネントと関連付けられている JSP の同期を維持します。ページコンポーネントに子の可視コンポーネントが追加されると、関連付けられているすべての JSP に自動的に JSP タグを追加します。同様に、子の可視コンポーネントが名前変更されたり、削除されたりすると、設計時の同期を維持するために自動的にタグを削除したり、名前を変更したりします。たとえば「foo」という名前の「テキストフィールド」を子の可視コンポーネントに追加すると、「JSP ページ」ノードの一覧にある、関連付けられているすべての JSP に `jato:textField` タグが挿入されます。

この同期は一方向です。ページコンポーネントに対する変更は自動的に JSP に反映されますが、その逆の反映は行われません。ページコンポーネントが主であり、JSP は従です。JSP は標準的な「利用」関係でページコンポーネントを利用します。

正式には各 JSP は 1 つの Web アプリケーションフレームワークページにだけ関連付けることができます。ただし、Web アプリケーションフレームワークページには、ページレットの子の可視コンポーネントが含まれることがあるため、ページおよびページレット混在の任意の階層を簡単に作成して、アプリケーションレベルの可視コンポーネントを最大限に再利用することができます。

関連付けられた JSP には、以下に示すようなページコンポーネントのクラス名を示す jato:useViewBean タグが付きます。

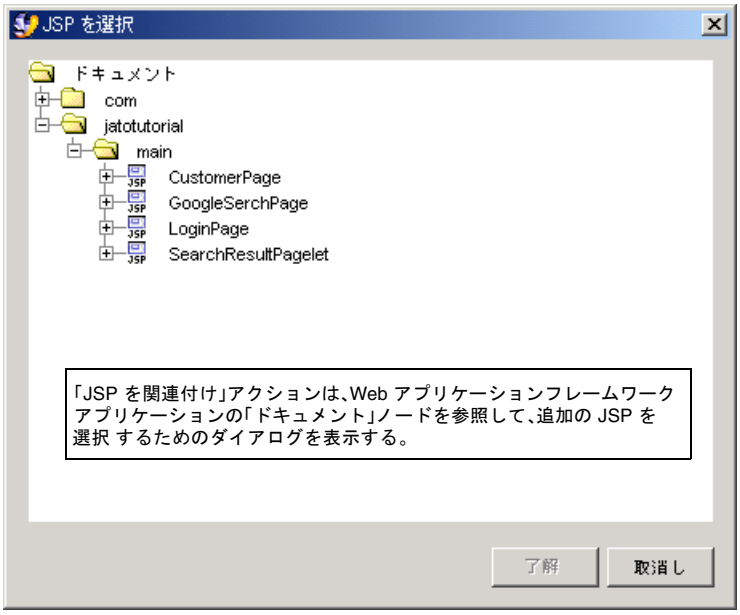
```
<jato:useViewBean className="jatotutorial.main.CustomerPage">
```

「JSP ページ」ノードのコンテキストメニューコマンド

▼ [アクション]「JSP を関連付け」

このアクションは、ページコンポーネントに関連付ける JSP を追加します。現在のアプリケーションドキュメントルートにある使用可能な JSP の一覧から関連付ける JSP を選択します。「JSP を選択」ダイアログの「了解」ボタンは、その時点でどのページにも関連付けられていない JSP が選択された場合にのみ使用可能になります。このコマンドは関連付けられている JSP の内容を変更して、JSP が適切な Web アプリケーションフレームワークライブラリ記述子 (tld) 指令および、関連付けられている JSP とこのページ間の利用関係を宣言する Web アプリケーションフレームワーク JSP タグで構成されるようにします。

次の図は、「JSP を選択」エディタを示しています。



▼ [アクション]「JSP を追加」

このアクションは新しい JSP ページを作成して、このページに関連付けます。新しく生成された JSP には、このページおよびこのページの可視コンポーネントの子のすべてに対応する JSP タグが自動的に挿入されます。新しい JSP ページが作成されると、このページの、関連付けられている JSP ページ一覧に追加され、「JSP ページ」ノードの下に新しい JSP サブノードが表示されます。

▼ [アクション]「順序を変更」

このアクションは、現在のノードのサブノードの一覧をダイアログで表示します。このダイアログには、サブノードの表示順序を操作するためのボタン (「上へ移動」「下へ移動」) があります。

「JSP ページ」ノードのプロパティ

▼ [プロパティ]「デフォルト JSP」

このプロパティは、関連付けられている JSP のうち、現在のページの「デフォルト JSP」として指定されている JSP の URI を示します。デフォルト JSP は、Sun Java Studio Enterprise 7 でこのページをテスト実行するときに、ページ固有のコードでの優先指定がない限り、現在のページの生成に使用される JSP です。このプロパティは編集できません。

「デフォルト JSP」プロパティの現在の設定値は、以下のようにページコンポーネントの生成された Java ソースコードに反映されます。

```
setDefaultDisplayURL("/jatotutorial/main/CustomerPage.jsp");
```

「JSP ページ」ノードのサブノード

JSP が関連付けられている場合、「JSP ページ」ノードに JSP サブノードが存在します。

JSP ノード

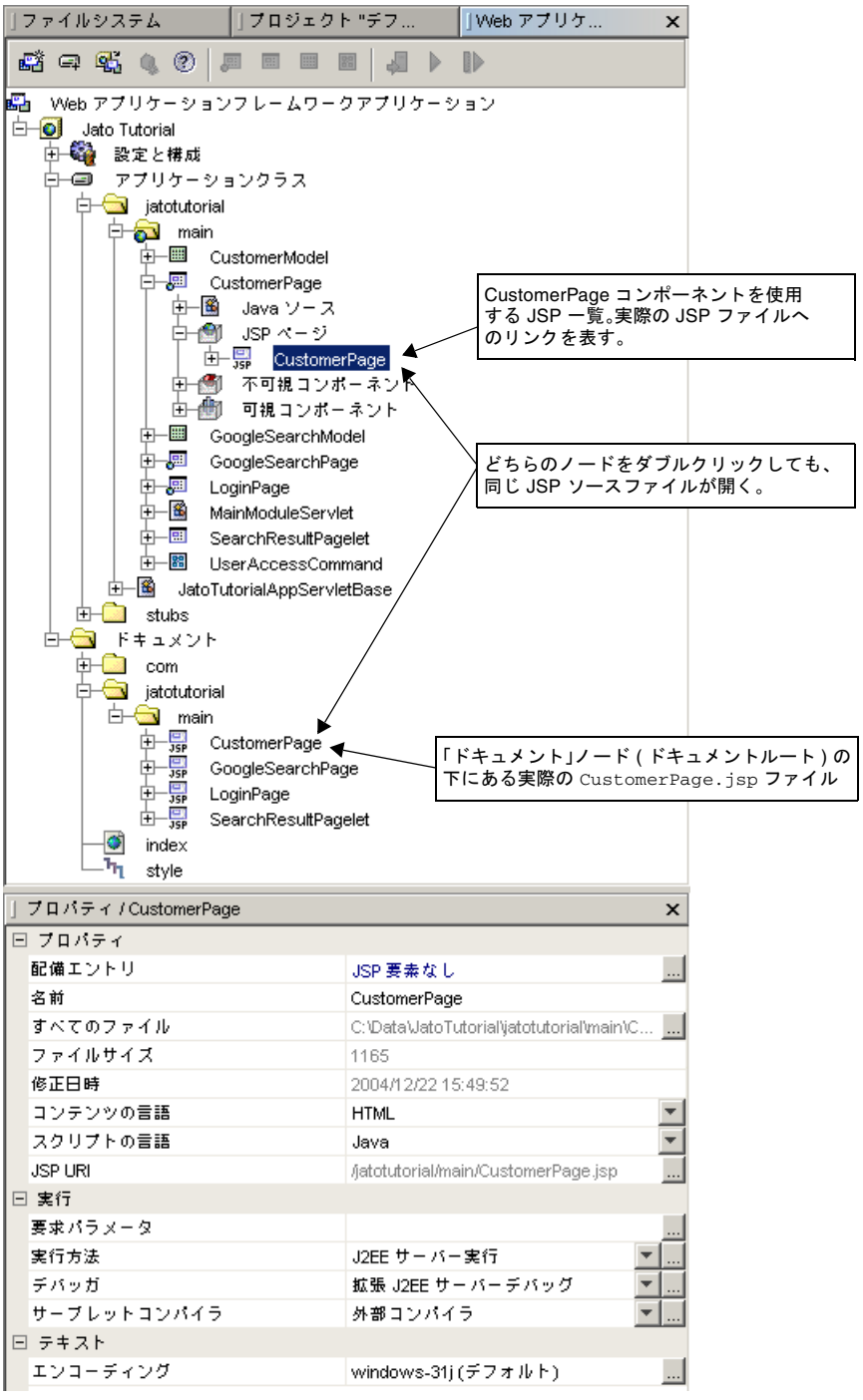
JSP ノードは、ページコンポーネントに関連付けられている JSP ファイルにノードレベルでアクセスできるようにします。JSP ノードは、Sun Java Studio Enterprise 7 の従来の JSP ノードと同じです。

物理的には JSP ファイルはアプリケーションのドキュメントルートディレクトリの下に位置し、ページコンポーネントはアプリケーションの `WEB-INF/classes` ディレクトリに位置しますが、IDE は JSP ノードの複製を作成し、Web アプリケーションフレームワークのページコンポーネント開発者が、そのページコンポーネントノードの階層内から、関連付けられている JSP に簡単にアクセスできるようにします。

これはノードの複製にすぎないことに注意してください。JSP ノードが 2 個所に現れているとしても、物理的に存在する JSP ファイルは 1 つだけです。JSP がページコンポーネントに関連付けられていること、また、その JSP が使用しているページコンポーネントに変更が加えられると、IDE によって自動的に JSP の内容の同期を取られることが視覚的に確認できます。

JSP ノードが 2 つのノード階層、すなわち、ドキュメントノード階層とページノード階層の両方に同時に存在することを見ることができます。

次の図は JSP ノードを示しています。


ファイルシステム
プロジェクト "デフ...
Web アプリケ...
Web アプリケーションフレームワークアプリケーション
Jato Tutorial
設定と構成
アプリケーションクラス
jatotutorial
main
CustomerModel
CustomerPage
Java ソース
JSP ページ
CustomerPage
不可視コンポーネント
可視コンポーネント
GoogleSearchModel
GoogleSearchPage
LoginPage
MainModuleServlet
SearchResultPagelet
UserAccessCommand
JatoTutorialAppServletBase
stubs
ドキュメント
com
jatotutorial
main
CustomerPage
GoogleSearchPage
LoginPage
SearchResultPagelet
index
style
CustomerPage コンポーネントを使用する JSP 一覧。実際の JSP ファイルへのリンクを表す。
どちらのノードをダブルクリックしても、同じ JSP ソースファイルが開く。
「ドキュメント」ノード (ドキュメントルート) の下にある実際の CustomerPage.jsp ファイル
プロパティ / CustomerPage
プロパティ
配備エントリ
JSP 要素なし
名前
CustomerPage
すべてのファイル
C:\Data\JatoTutorial\jatotutorial\main\C...
ファイルサイズ
1165
修正日時
2004/12/22 15:49:52
コンテンツの言語
HTML
スクリプトの言語
Java
JSP URI
/jatotutorial/main/CustomerPage.jsp
実行
要求パラメータ
実行方法
J2EE サーバー実行
デバッガ
拡張 J2EE サーバーデバッグ
サーブレットコンパイラ
外部コンパイラ
テキスト
エンコーディング
windows-31j (デフォルト)

「ドキュメント」ノードの下の JSP は実際の JSP ファイルを表しているのに対し、「JSP ページ」の下の JSP は実際のファイルへのリンクです。つまり、「JSP ページ」ノードの下の JSP を削除しても、Sun Java Studio Enterprise 7 内のページコンポーネントとの関連付けが削除されるだけです。ただし、「ドキュメント」ルートの下の JSP を削除した場合は、ディスクから物理的にファイルを削除することになります。

JSP ノードのコンテキストメニューコマンド

JSP ノードのコンテキストメニューコマンドは、以下に詳述するコマンドを除けば、Sun Java Studio Enterprise 7 の従来の JSP ノードのコンテキストメニューコマンドと同じです。他の Java ノードのコンテキストメニューコマンドとプロパティについては、Sun Java Studio Enterprise 7 のマニュアルに完全な説明があります。

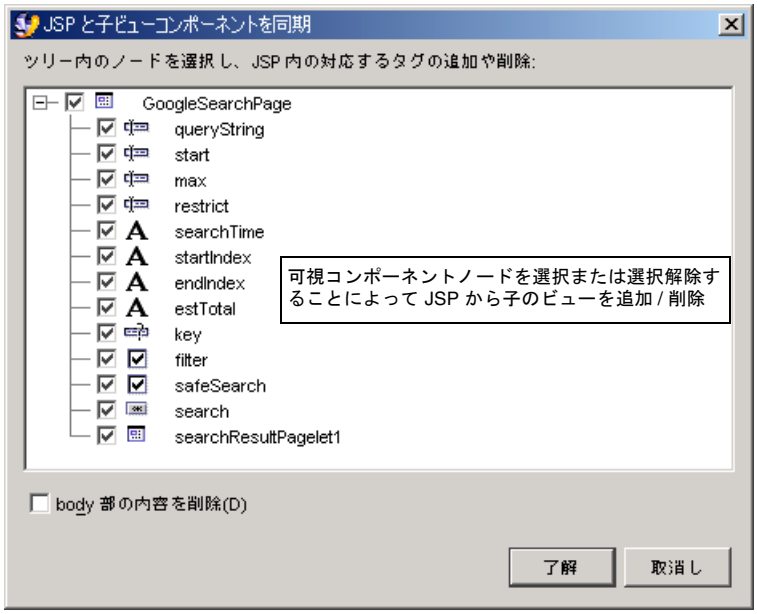
▼ [アクション]「ビューと同期」

このアクションは、ページの可視コンポーネント階層に対応する JSP タグを選択的に追加または削除することを可能にする同期ダイアログを表示します。このコマンドは単なる便利なユーティリティです。当然、JSP タグは、JSP を直接編集して追加したり、削除したりできます。ただし、「ビューと同期」は、バッチモードでタグを追加したり削除したりする便利な手段を提供します。

このエディタは、ページコンポーネントの可視コンポーネント階層全体を 1 つの簡単なノードツリー形式で提供します。ページコンポーネントの「可視コンポーネント」ノードに存在する子のすべてがサブノードとして表示されます。複雑な可視コンポーネント階層になると、子の可視コンポーネントを持つページレット (たとえば TiledView) が子として含まれることがあります。このエディタでは、どのようにビュー階層が深くて複雑でも、階層全体が表示されます。

ページに含まれていて、JSP 内のタグで現在参照されている可視コンポーネントはすべて、選択状態 (チェックマークが付いている) で表示されます。ページに含まれていて、JSP 内のタグで現在参照されていない可視コンポーネントはすべて、選択解除状態 (チェックマークが付いていない) で表示されます。さまざまなノードを選択または選択解除することによって、JSP に一括変更を加えることができます。

次の図は同期ダイアログを示しています。



不要なタグは、対応するノードを選択解除することによって削除することができます。タイルビューのように子がページレットの場合、子を選択解除するとそのすべての子も選択解除され、すべて削除されます。

タグの追加は、対応するノードを選択して行います。子としてのページレットを選択すると、そのすべての子も選択され、必要に応じて入れ子になっている任意の子を選んで選択状態を解除できます。

同期ダイアログの最下部にある「body 部の内容を削除」チェックボックスでは、本体に内容を持つタグ (HREF や Pagelet タグなど) に対するタグ削除ポリシーを制御できます。本体の内容も削除するか、あるいはタグは削除しても本体の内容はそのまま残すかを選択することができます。デフォルトでは「body 部の内容を削除」チェックボックスは選択解除されており、ユーザーの同意なしに既存の本体の内容を削除できないようになっています。

▼ [アクション]「デフォルトとして設定」

このアクションは、現在の JSP をページコンポーネントに対するデフォルト JSP に設定します。

▼ [アクション]「削除」

このアクションは「JSP ページ」ノードから現在の JSP ノードを削除し、JSP とページコンポーネント間の正式な関連付けを解除します。ただし、このコマンドは、ディスクから物理的に JSP ファイルを削除するわけではありません。JSP ファイルと現在

のページコンポーネント間の関連付けを削除するだけです。JSP ファイルそのものを物理的に削除するには、ドキュメントルートの下にある実際の JSP ノードを削除します。

JSP ノードのプロパティ

JSP ノードのプロパティは、以下に詳述するプロパティを除けば、Sun Java Studio Enterprise 7 の従来の JSP ノードのプロパティと同じです。他の Java ノードのプロパティについては、Sun Java Studio Enterprise 7 のマニュアルに完全な説明があります。

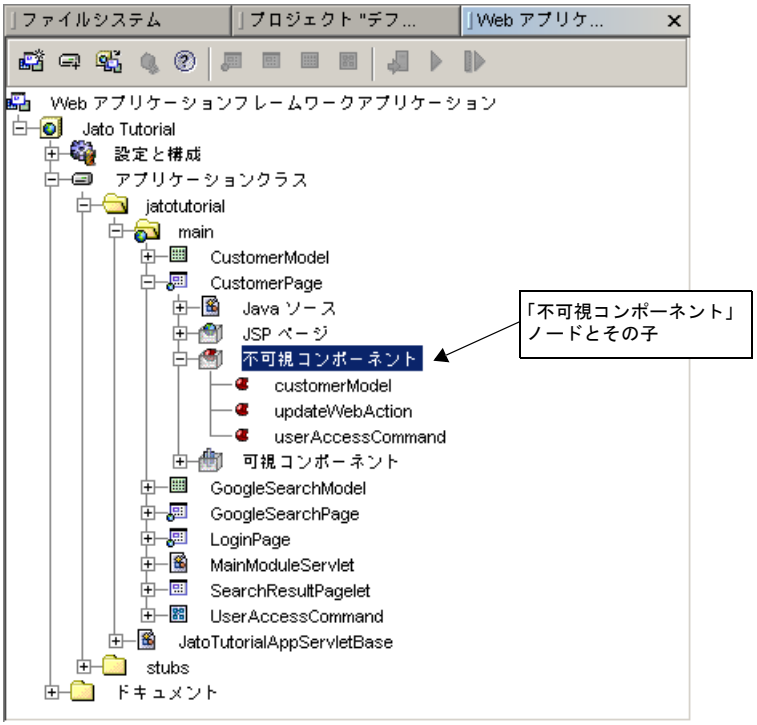
▼ [プロパティ]「JSP URI」

このプロパティは、アプリケーションのドキュメントルートの下の JSP ファイルの場所を示します。これは、単に JSP ファイルの物理的な場所を一目で分かるようにするためのプロパティです。このプロパティは編集できません。

「不可視コンポーネント」ノード

不可視コンポーネントは、ページコンポーネントが使用する不可視のコンポーネントです。たとえば、ページコンポーネントはモデルやコマンドコンポーネントを「利用」したり、「参照」したりします。そうしたコンポーネントを 1 つも使用、参照しないこともあります。こうした参照は、不可視コンポーネントノードとして宣言することによって作成できます。コンポーネントライブラリもまた、利用の目的で任意の不可視コンポーネントを作成します。IDE は、宣言によるこうした不可視のリソースの作成に合わせて対応する Java コードを生成します。

次の図は「不可視コンポーネント」ノードとその子を示しています。



本質的にはこれらの不可視コンポーネントは単なる JavaBean であるため、ここに表示できるコンポーネントの種類には限りがなく、このため、コンポーネントの種類によってプロパティシートは異なったものになります。一般に、不可視コンポーネントを定義するのは、値を設定して、不可視コンポーネントを参照する必要があるプロパティが、ページの可視コンポーネントに存在するためです。この内部「利用」関係の古典的な例としては、不可視の `ModelReference` コンポーネントを参照することによって満たされる可視コンポーネントの「`ModelReference`」プロパティがあります。可視コンポーネントのプロパティエディタの中には、そうしたプロパティが設定されたときに暗黙で不可視コンポーネントを作成するものがあります。ページに明示的に不可視コンポーネントを追加することもできます。

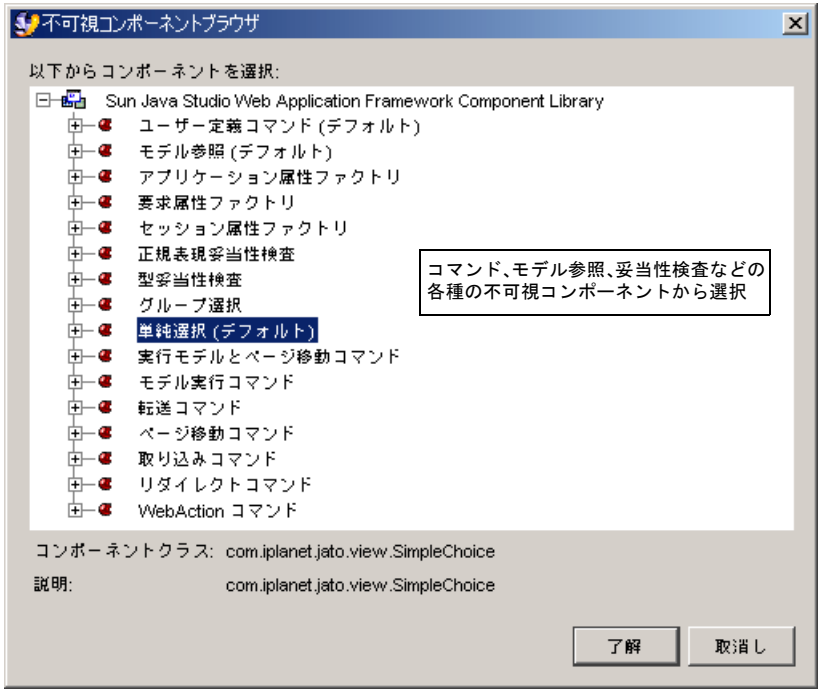
不可視コンポーネントは、現在のページ内の可視コンポーネントによって使用されることもあれば、使用されないこともあります。

「不可視コンポーネント」ノードのコンテキストメニューコマンド

▼ [アクション]「不可視コンポーネント...追加」

このアクションは明示的に新しい不可視コンポーネントを追加します。Web アプリケーションフレームワークアプリケーションに登録されている Web アプリケーションフレームワークコンポーネントライブラリで使用可能な不可視コンポーネントの一覧から選択してください。

次の図は不可視コンポーネントブラウザを示しています。



▼ [アクション]「順序を変更」

このアクションは、現在のノードのサブノードの一覧をダイアログで表示します。このダイアログには、サブノードの表示順序を操作するためのボタン (「上へ移動」「下へ移動」) があります。

「不可視コンポーネント」ノードのプロパティ

▼ [なし]

「不可視コンポーネント」ノードのサブノード

不可視コンポーネントが存在する場合、「不可視コンポーネント」ノードは不可視コンポーネントサブノードから構成されます。

不可視コンポーネントのノード

不可視コンポーネントノードは、現在のページのスコープ内でインスタンス化されている不可視コンポーネントを宣言することによって作成します。IDE はページの Java ソースファイル内に不可視コンポーネントをインスタンス化するためのコードを自動的に生成します。

不可視コンポーネントノードのコンテキストメニューコマンド

▼ [アクション]「カット」、「コピー」、「ペースト」

従来のカット、コピー、ペースト動作をするアクションです。

▼ [アクション]「削除」

このアクションは不可視コンポーネントノードを削除します。不可視コンポーネントを削除すると、その不可視コンポーネントに対する参照を保持するプロパティ値を持つすべての可視コンポーネントに警告の印が付けられます。後で整理できるよう、ページノードで提供されるエラー情報には、古くなったプロパティ参照に関する詳細な情報が含まれます。

不可視コンポーネントノードのプロパティ

1 つの不可視コンポーネントには、3 つのプロパティタブがあります。

- プロパティ
- 「コンポーネントプロパティ」
- 「コード生成」

「プロパティ」と「コード生成」タブには、どの不可視コンポーネントにも同じプロパティが含まれています (この後のプロパティの説明を参照)。

これに対し「コンポーネントプロパティ」ウィンドウには、それぞれの不可視コンポーネントの種類に固有のプロパティが含まれています。不可視コンポーネントは任意の JavaBean であるため、IDE によって各 Bean に専用のプロパティが 動的に検出されて表示されます。それらのプロパティは、この「コンポーネントプロパティ」ウィンドウで編集することができます。不可視コンポーネントに固有のプロパティはコンポーネントによって異なるため、ここでは説明しません。

▼ [プロパティ]「コンポーネントクラス」

このプロパティは、不可視コンポーネントの完全限定クラス名を示します。このプロパティは編集できません。

▼ [プロパティ]「名前」

このプロパティは、不可視コンポーネントの論理名を示します。論理名とは、このページコンポーネントのスコープ内で不可視コンポーネントが可視コンポーネントのプロパティによって参照されているあらゆる場所に表示される名前です。IDE は生成するコードの一部としてこの名前のプロパティ値を使用し、ページの Java ソースファイル内での不可視コンポーネントの作成およびアクセスを管理します。生成される正確なコード内容は、関係する 3 つのコード生成プロパティの値によって異なります。

- アクセス権
- Scope
- セッション属性名

▼ [プロパティ]「アクセス権」

このプロパティは、ページの Java ソース内で不可視コンポーネントインスタンスにアクセスできるようにするコードを生成する際に、IDE のコード生成エンジンが適用するアクセス権限を示します。

▼ [プロパティ]「スコープ」

このプロパティは、ページの Java ソース内で不可視コンポーネントインスタンスにアクセスできるようにするコードを IDE のコード生成エンジンが生成する際にエンジンの動作を制御することを可能にするスコープ設定を示します。

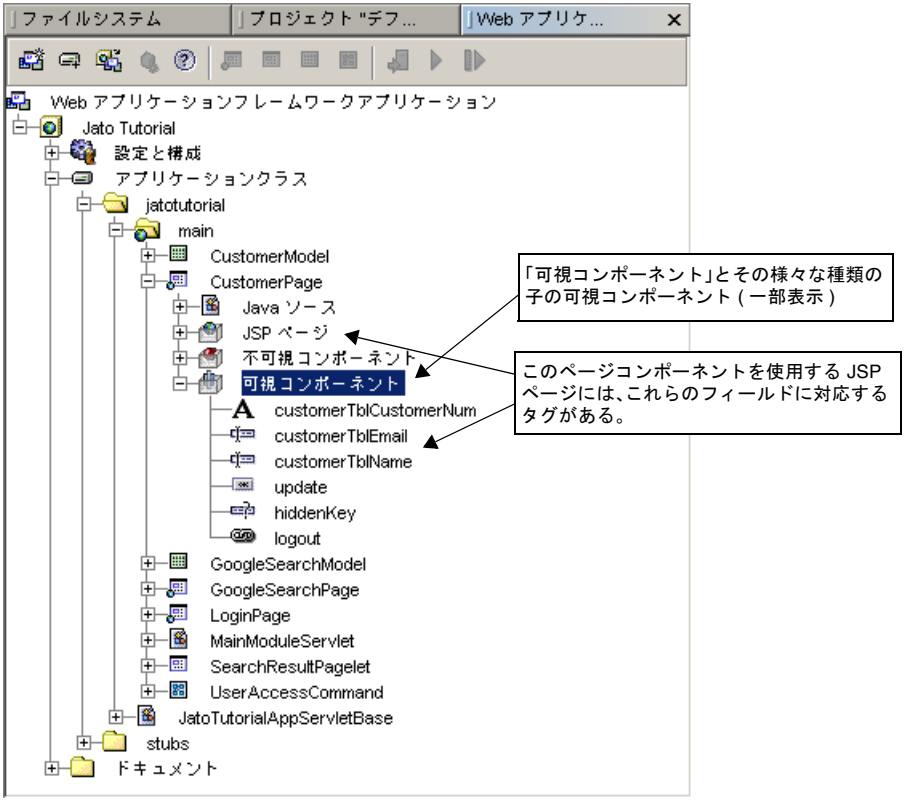
▼ [プロパティ]「セッション属性名」

このプロパティは、HTTP セッションの属性名を示します。「スコープ」属性の設定が HTTP セッションの場合、コード生成エンジンは、ページの Java ソース内で不可視コンポーネントインスタンスにアクセスできるようにするコードを生成する際に、指定されたセッション属性名を使用します。

「可視コンポーネント」ノード

可視コンポーネントが存在する場合、「可視コンポーネント」ノードには子の可視コンポーネントノードが含まれます。これは、可視コンポーネントについての核心部分です。ページノードは階層のルートを形成します。可視コンポーネントが存在する場合、ページノードには可視コンポーネントサブノードが含まれます。ページレットコンポーネントである可視コンポーネントサブノードは、それ自体が子の可視コンポーネントを含むことができます。このため、任意の深さの可視コンポーネント階層が可能になります。

次の図は「可視コンポーネント」ノードとその様々な種類の子の可視コンポーネントを示しています。



「可視コンポーネント」ノードのコンテキストメニューコマンド

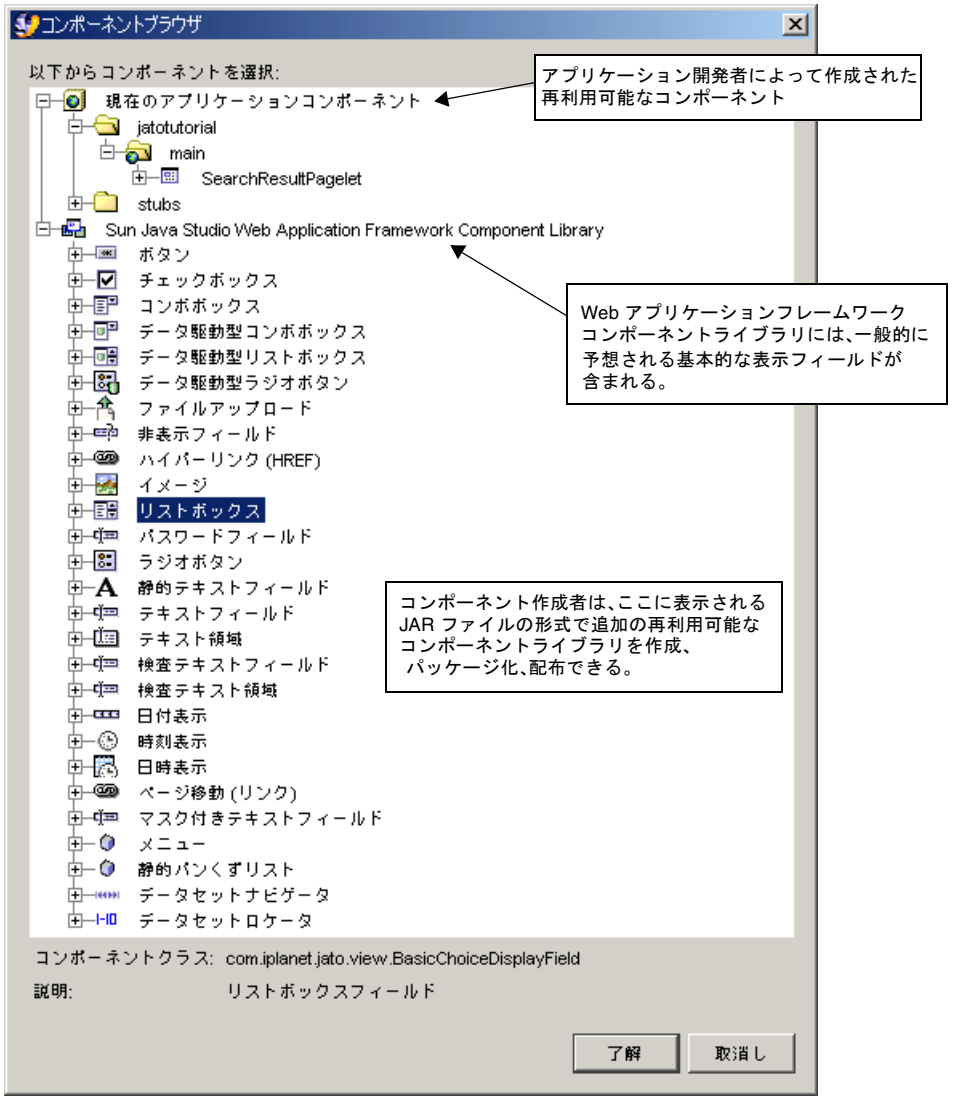
▼ [アクション]「可視コンポーネント...追加」

このアクションは、新しい可視コンポーネントノードを追加します。このために、このコマンドはコンポーネントブラウザを表示します。コンポーネントブラウザからコンポーネントを選択するのは、「コンポーネント」パレットをクリックすることによって可視コンポーネントを追加するのと機能的に等価です。これらは、可視コンポーネントを追加する等しく有効な手段です。

コンポーネントブラウザは、少なくとも、「現在のアプリケーションコンポーネント」と「Web アプリケーションフレームワーク Component Library」という 2 つのノードが入った状態で表示されます。

Sun 以外の Web アプリケーションフレームワークコンポーネントライブラリの JAR ファイルを `WEB-INF/lib` ディレクトリに追加した場合は、そのコンポーネントライブラリを表すノードも表示されます。再利用可能、配布可能な独自の Web アプリケーションフレームワークコンポーネントライブラリの作成についての詳細は、『Web アプリケーションフレームワーク コンポーネント作成ガイド』をご覧ください。

次の図はコンポーネントブラウザを示しています。



この方法でページに追加されたどの可視コンポーネントノードに対しても、IDE はコンポーネントコンストラクタ呼び出しとコンポーネント補助メソッド、コンポーネントインスタンス名のクラス定数を生成します。また、対応するコンポーネントに固有の JSP タグを、そのページコンポーネントを使用するすべての JSP に挿入します。

これらの自動処理によって、矛盾がなくエラーを起こしにくいアプリケーションの実現が容易になります。もっと柔軟な可視コンポーネントの作成が求められる場合は、手動作成という選択肢があります。『Web アプリケーションフレームワーク 開発ガイド』をご覧ください。

▼ [アクション]「ペースト」

可視コンポーネントをペーストします。別のページからコピーまたはペーストした可視コンポーネントをペーストした場合、元の可視コンポーネントノードによって参照されている不可視コンポーネントが自動的に新しいページに移されることはありません。

▼ [アクション]「順序を変更」

このアクションは、現在のノードのサブノードの一覧をダイアログで表示します。このダイアログには、サブノードの表示順序を操作するためのボタン (「上へ移動」「下へ移動」) があります。

可視コンポーネントノードのプロパティ

▼ [なし]

「可視コンポーネント」ノードのサブノード

可視コンポーネントが存在する場合、「可視コンポーネント」ノードは可視コンポーネントサブノードで構成されます。

可視コンポーネントノード

可視コンポーネントノードは単純な可視コンポーネントノード (テキストフィールド、選択フィールド、コマンドフィールドなど) のこともあれば、ページレットノード (タイルビュー、コンテナビュー、ツリービュー) のこともあります。ページレットノードは可視コンポーネントサブノードに展開可能で、それ自体任意の可視コンポーネント階層を持つことができます。

可視コンポーネントノードのコンテキストメニューコマンド

▼ [アクション]「イベント」

このアクションは、親コンポーネントのソースコードファイルでの実装に適した、子に固有のイベントハンドラメソッドの一覧からなるサブメニューを起動します。すでに実装されているイベントハンドラの名前は太字で表示されます。実装されていないメソッドの名前は太字で表示されません。一覧から太字のイベントハンドラを選択すると、IDE が親コンポーネントの Java ソースを開き、そのイベントハンドラの先頭にカーソルを移動します。太字でないイベントハンドラを選択すると、IDE が親コン

ポーネントの Java ソースファイルに、対応するイベントハンドラのメソッドスタブを追加し、その上にカーソルを移動します。イベントハンドラが子コンポーネント専用である必要があるため、IDE はイベントハンドラメソッドを生成し、子コンポーネントのインスタンス名に基づいてメソッドが一意の名を持つようにします。たとえば「イベント」メニューでイベントハンドラが「beginDisplay」と表示され、可視コンポーネントのインスタンス名が「foo」の場合、IDE は「beginFooDisplay」という名前のハンドラメソッドを生成します。

使用可能な子コンポーネントのイベントハンドラメソッドの一覧の内容は、親コンポーネントの子コンポーネント固有のイベントをディスパッチする親コンポーネントの能力によって決まります。この能力は、現在のページコンポーネントのもとになっているコンポーネントによって宣言されています。現在の子コンポーネントの種類に対するイベントハンドラがページコンポーネントによって宣言されていない場合、可視コンポーネントノードのイベントのコンテキストメニューコマンドは使用不可になります。

可視コンポーネントノードのプロパティ

あらゆる可視コンポーネントノードが、複数の共通プロパティを共有します。また、可視コンポーネントはそれぞれに、IDE が動的に検出し、提示する可視コンポーネント固有の任意の個数のプロパティがあり、それらプロパティはプロパティシートで設定できます。

▼ [プロパティ]「コンポーネントクラス」

このプロパティは、この可視コンポーネントの完全限定クラス名を示します。このプロパティは編集できません。

▼ [プロパティ]「ComponentInfo クラス」

このプロパティは、この可視コンポーネントの完全限定 `ComponentInfo` クラスを示します。このプロパティは編集できません。

▼ [プロパティ]「名前」

可視コンポーネントのインスタンス名を示します。IDE は生成するコードの一部としてこの名前のプロパティ値を使用し、ページの Java ソースファイル内での可視コンポーネントの作成およびアクセスを管理します。

▼ [プロパティ]「初期化前コード」

このプロパティは、可視コンポーネントのコンストラクタの呼び出し直後に、コード生成エンジンが挿入する任意の Java コードブロックを示します。このため、この初期化前コードは、明示的なプロパティ設定機能が呼び出される前に挿入されます。この機能を利用して、ページの Java ファイル内の他の方法で編集できない部分にコードを挿入することができます。

▼ [プロパティ] 「初期化後コード」

このプロパティは、自動的に生成された可視コンポーネント固有のプロパティ設定呼び出しコードの直後に、コード生成エンジンが挿入する任意の Java コードブロックを示します。この機能を利用して、ページの Java ファイル内の他の方法で編集できない部分にコードを挿入することができます。

ページレットコンポーネントのノード

実質的には、ページレットノードはページノードと同じです。両者とも類似のサブノード、コンテキストメニューコマンド、プロパティを持ちます。1 つ注目すべき違いは、ページレットノードはページノードのように実行できない点です。これは、ページレットがサブページ、つまり、ページの断片であるためです。ページレットの目的は、他のもっと複雑な可視コンポーネントの基礎となる、目に見えるブロックを提供し、最終的にページコンポーネントの子として直接または間接にそれらブロックが含まれるようにすることにあります。ページコンポーネントは、可視コンポーネント階層の根を形成します。これに対しページレットは、可視コンポーネント階層の枝を形成します。ページレットは、その子として他のページレットを含むことができます。ページレットは、任意の数の他のページレットまたはページコンポーネントの子になることができます。

たとえばタイルビューコンポーネントはページレットの 1 つです。タイルビューには、繰り返し動作が組み込まれており、結果セット全体を繰り返して、各行の列 (タイル) の値を表示する目的によく使用されます。

ページレットコンポーネントノードのコンテキストメニューコマンド

実行コマンドがないことを除けば、ページレットのコンテキストメニューコマンドはページノードのコンテキストメニューコマンドと同じです。ページレットをテスト実行することはできません。ページのように、ページレットを URL でアドレス設定することはできません。

ページコンポーネントノードのコンテキストメニューコマンドを参照してください。

ページレットコンポーネントノードのプロパティ

ページコンポーネントノードのコンテキストメニューコマンドを参照してください。

ページレットコンポーネントのサブノード

ページコンポーネントノードのコンテキストメニューコマンドを参照してください。

ページレット JSP ノードに関する特記事項

ページレットノードがどの JSP ノードにも関連付けられていないことは完全に許容されることです。Web アプリケーションフレームワークページレットがまずもって可視コンポーネント階層であることを理解するまでは、このことは直感に反しているように聞こえるかもしれません。可視コンポーネント階層の生成は、対応する JSP タグによって制御されますが、JSP タグをページレット別の JSP の断片にする必要はありません。

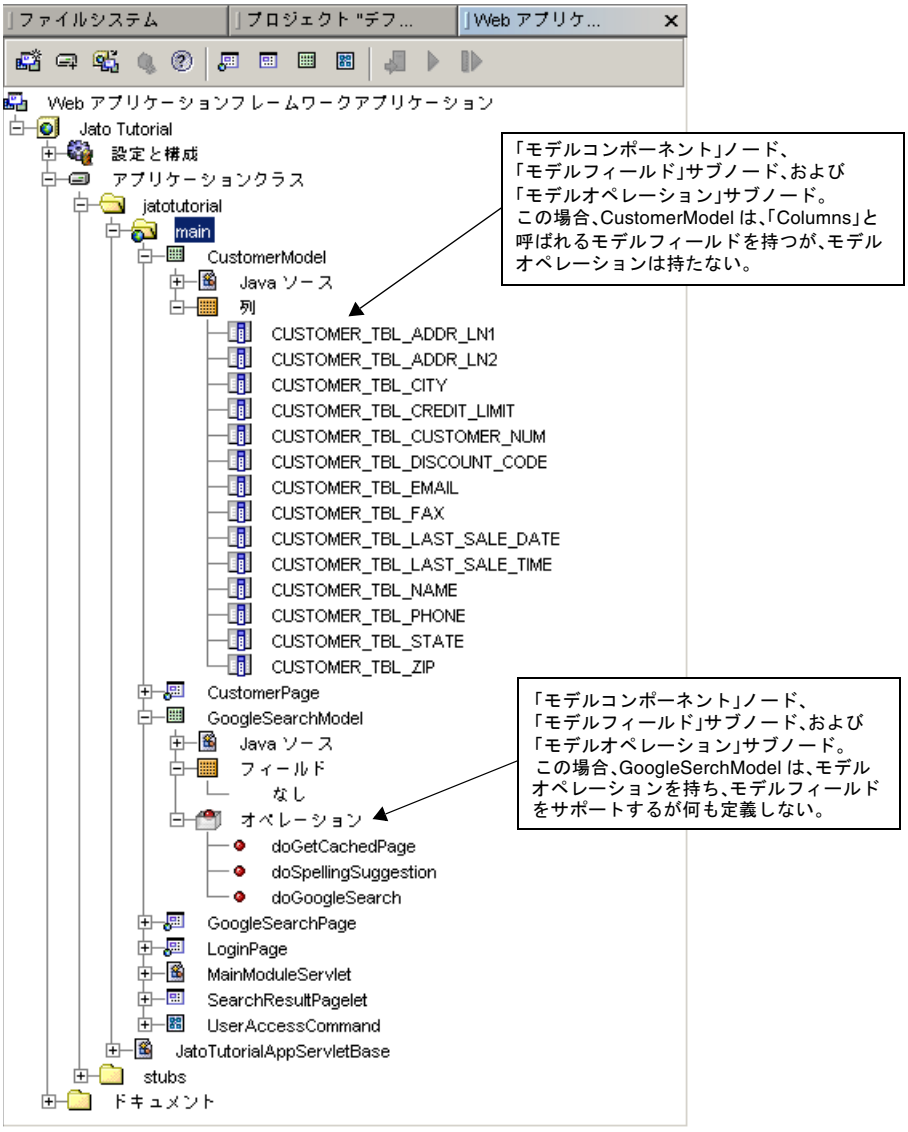
古典的なページレットの使用例では、ページレット専用の可視コンポーネント階層を生成するためのタグは、再利用可能な JSP 断片単位であり、JSP サブノードとして正式にページレットコンポーネントに関連付けられます。その場合、ページレットが別のコンテナに子として追加されるたびに、IDE は親の可視コンポーネントの JSP ファイルに JSP インクルードタグを 1 つ追加します。このインクルードタグには、従来の JSP インクルードのやり方でページレットの JSP 断片が含まれます。これは、ページレットが子として現れるあらゆるコンテキストでページレットの生成を同じにする場合に取るべき方法です。

ただし、ケースによっては、ページレットを使用する親コンポーネントが異なるのに応じて、そのページレットの可視レイアウトが変わるようにしたい場合もあります。その場合は、意図的にそのページレットに明示的に JSP を関連付けることを控えることになるでしょう。この場合は、別の可視コンポーネントに子としてページレットが追加されるたびに、IDE は親の可視コンポーネントの JSP ファイルにページレットとその子用のタグを追加します (こうして、親の JSP ファイルに深いタグ階層が作成される)。この機能によって、親の JSP を 1 つのドキュメントとして編集することが可能になり、JSP 作成者にはこの方法を取る人もいます。Web アプリケーションフレームワークにはこの選択肢が用意されており、ページレット別に決定することができます。

モデルコンポーネントノード

モデルは、アプリケーションのビュー用のデータ (ページおよびページレット) に対する記憶機構またはプロキシ、あるいはその両方の働きをするクラスです。Web アプリケーションフレームワークには、JDBC SQL 照会、JDBC ストアドプロシージャ、Web サービス、リソースバンドル、Bean アダプタ、カスタム、HTTP セッション、カスタムツリー、オブジェクトアダプタなど、いくつもの種類のモデルが付属しています。

次の図はモデルコンポーネントノードの例を示しています。


ファイルシステム
プロジェクト "デフ...
Web アプリケ...
Web アプリケーションフレームワークアプリケーション
Jato Tutorial
設定と構成
アプリケーションクラス
jatotutorial
main
CustomerModel
Java ソース
列
CUSTOMER_TBL_ADDR_LN1
CUSTOMER_TBL_ADDR_LN2
CUSTOMER_TBL_CITY
CUSTOMER_TBL_CREDIT_LIMIT
CUSTOMER_TBL_CUSTOMER_NUM
CUSTOMER_TBL_DISCOUNT_CODE
CUSTOMER_TBL_EMAIL
CUSTOMER_TBL_FAX
CUSTOMER_TBL_LAST_SALE_DATE
CUSTOMER_TBL_LAST_SALE_TIME
CUSTOMER_TBL_NAME
CUSTOMER_TBL_PHONE
CUSTOMER_TBL_STATE
CUSTOMER_TBL_ZIP
CustomerPage
GoogleSearchModel
Java ソース
フィールド
なし
オペレーション
doGetCachedPage
doSpellingSuggestion
doGoogleSearch
GoogleSearchPage
LoginPage
MainModuleServlet
SearchResultPagelet
UserAccessCommand
JatoTutorialAppServletBase
stubs
ドキュメント
「モデルコンポーネント」ノード、「モデルフィールド」サブノード、および「モデルオペレーション」サブノード。この場合、CustomerModel は、「Columns」と呼ばれるモデルフィールドを持つが、モデルオペレーションは持たない。
「モデルコンポーネント」ノード、「モデルフィールド」サブノード、および「モデルオペレーション」サブノード。この場合、GoogleSerchModel は、モデルオペレーションを持ち、モデルフィールドをサポートするが何も定義しない。

モデルコンポーネントノードのコンテキストメニューコマンド

▼ [アクション]「開く」

このアクションはソースエディタでコンポーネントの Java ソースファイルを開いて、そのクラスにジャンプします。

▼ [アクション]「イベント」

このアクションは、このコンポーネントへの実装に適したイベントハンドラメソッドの一覧からなるサブメニューを起動します。すでに実装されているイベントハンドラの名前は太字で表示されます。実装されていないメソッドの名前は太字で表示されません。一覧から太字のイベントハンドラを選択すると、IDE がコンポーネントの Java ソースを開き、そのイベントハンドラの先頭にカーソルを移動します。太字でないイベントハンドラを選択すると、IDE がコンポーネントの Java ソースファイルに、対応するイベントハンドラのメソッドスタブを追加し、その上にカーソルを移動します。利用可能なイベントハンドラメソッドの一覧はコンポーネントによって異なり、現在のコンポーネントのもとになっているコンポーネントによって宣言されています。コンポーネントによっては、イベントハンドラを宣言していないものもあります。コンポーネントがイベントハンドラを 1 つも宣言していない場合、このメニューコマンドは使用不可になります。

▼ [アクション]「デザインアクション」

このアクションは、このコンポーネントで使用可能な特製の設計アクションの一覧からなるサブメニューを起動します。設計アクションとは、コンポーネントの構成を操作または変更する機能を持つ任意の関数です。この関数は、コンポーネントの作成者が提供します。設計アクションは何もしないことでもかまいませんし、コンポーネントのカスタマイザを表示することもできます。

▼ [アクション]「コンポーネントメソッド」

このアクションは、このコンポーネントが実装可能な、有用なコンポーネントメソッドの一覧からなるサブメニューを起動します。すでに実装されているコンポーネントメソッドの名前は太字で表示されます。実装されていないメソッドの名前は太字で表示されません。一覧から太字のコンポーネントメソッドを選択すると、IDE がコンポーネントの Java ソースを開き、そのメソッドの先頭にカーソルを移動します。太字でないコンポーネントメソッドを選択すると、IDE がコンポーネントの Java ソースファイルに対応するメソッドスタブを追加し、その上にカーソルを移動します。

利用可能なコンポーネントメソッドの一覧はコンポーネントによって異なり、現在のコンポーネントのもとになっているコンポーネントによって宣言されています。コンポーネントによっては、コンポーネントメソッドを宣言していないものもあります。一般にコンポーネントメソッドは、利用可能なベースクラスメソッドのサブセットです。そのサブセットは、特にサブクラスによるオーバーライドを意図したもので、ラ

イブラリコンポーネントの作成者がこの方法を使用して公開することに意味があるものです。コンポーネントがイベントハンドラを 1 つも宣言していない場合、このメニューコマンドは使用不可になります。

▼ [アクション]「名前を変更」

コンポーネントクラス名を設定して、コンポーネントの Java ソースファイルの名前を変更します。

▼ [アクション]「削除」

コンポーネントを削除します。コンポーネントの Java ソースファイルも削除します。

▼ [アクション]「カット」、「コピー」、「ペースト」

従来のカット、コピー、ペースト動作をするアクションです。

モデルコンポーネントノードのプロパティ

▼ [プロパティ]「コンポーネントクラス」

コンポーネントの完全限定クラス名を示します。このプロパティは編集できません。

▼ [プロパティ]「ComponentInfo クラス」

コンポーネントの完全限定 ComponentInfo クラスを示します。このプロパティは編集できません。

▼ [プロパティ]「名前」

コンポーネントクラス名を設定します。

モデルコンポーネントのサブノード

モデルコンポーネントのサブノードは、モデルの種類によって少し異なることがあります。ページおよびページレットノード同様、あらゆるモデルは「Java ソース」サブノードを 1 つ持ちます。モデルコンポーネントはモデルフィールドグループノードを含むことができ、またモデルオペレーショングループノードもサポートします。

論理的には、オプションのモデルフィールドグループノードは一群の名前付きモデルフィールドを表します。画面上、モデルフィールドグループノードはモデルフィールドノードの親です。モデルフィールドの種類は任意で、コンポーネントの作成者が決定します。モデルフィールドグループノードのラベルは、そのノードが表すモデルフィールドの種類によって異なります。

すべてのモデルフィールドグループにまたがるモデルフィールドはどれも一意の名前を持つ必要があります。言い替えれば、モデル全体で一意の名前を持つモデルフィールドは 1 つ存在しますが、コンポーネントの作成者はそれらモデルフィールドを複数のグループに分けて、さまざまな種類のフィールドがサポートされるようにすることができます。

モデルにモデルフィールドグループがなくてもかまいません。その場合、ビューは、手動で入力された匿名バインドで、モデルにバインドされます。モデルにモデルフィールドノードあるいはモデルフィールドグループノードがなくても、バインド選択機能を使用して、内部的に生成された自動フィールドだけをモデルに提示させることもできます。こうしたことのすべてが、コンポーネントの作成者が下した判断およびモデルが管理しているデータの種類に依存します。数百、数千のバインドを示すモデルコンポーネントを作成することができます。その場合、すべてのフィールドをノードとして表すことは望ましいことではなく、コンポーネントの作成者がビューバインド用の特別な選択機能を提供することを推奨します。

論理的には、オプションのモデルオペレーショングループノードは、一群の名前付きモデルオペレーションを表します。画面上、モデルオペレーショングループノードはモデルオペレーションノードの親です。モデルオペレーショングループノードのラベルは、そのノードが表すモデルオペレーションの種類によって異なります。ただし、通常ラベルは「オペレーション」です。

「Java ソース」ノード

「Java ソース」ノードは、このコンポーネントの Java ソースファイルにノードレベルでアクセスできるようにします。Web アプリケーションフレームワークコンポーネントのノードは 「Java ソース」ノードの親になり、アプリケーションコンポーネントが論理的に単なる Java ノード 以上のものから構成されていることを強調します。以下の点を除けば、「Java ソース」ノードは、Sun Java Studio Enterprise 7 の従来の Java ソースコードのノードと同じです。

- Web アプリケーションフレームワークコンポーネントのノードが親である。
- ノード名は固定で、「Java ソース」である。
- コンテキストメニューの「カット」、「削除」、「名前を変更」コマンドは使用不可になる。これらのコマンドについては、コンテキストメニューではなく、コンポーネントのカット、削除、名前を変更コマンドを使用します。

「Java ソース」ノードのコンテキストメニューコマンドとプロパティについては、Sun Java Studio Enterprise 7 のマニュアルに完全な説明があります。

モデルフィールドグループノード

モデルフィールドグループノードは、一群のモデルフィールドノードの親ノードです。

モデルフィールドグループノードのコンテキストメニューコマンド

▼ [アクション]「フィールド...追加」

グループに新しいモデルフィールドを追加します。モデルフィールド用の新しいサブノードが追加されます。追加されたフィールドのデフォルト名は、モデルコンポーネントの作成者が定義したベース名にもとづく一意の名前です。開発者は、必要に応じて名前を変更できます。モデルフィールドが追加されると、モデルの Java ファイルの生成コードの部分が更新されます。

▼ [アクション]「ペースト」

モデルフィールドをペーストします。

▼ [アクション]「順序を変更」

このアクションは、現在のノードのサブノードの一覧をダイアログで表示します。このダイアログには、サブノードの表示順序を操作するためのボタン (「上へ移動」「下へ移動」) があります。

モデルフィールドグループノードのプロパティ

▼ [なし]

モデルフィールドグループノードのサブノード

モデルフィールドグループノードには、モデルフィールドサブノードが含まれます。

モデルフィールドノード

モデルフィールドノードでは、モデルフィールドのプロパティおよび編集アクションに直接アクセスできます。

モデルフィールドノードのコンテキストメニューコマンド

▼ [アクション]「カット」、「コピー」、「ペースト」

従来のカット、コピー、ペースト動作をするアクションです。モデルフィールドを別のモデルに移動すると、そのフィールドにバインドされていたすべての可視コンポーネントのバインドが解除されます。

▼ [アクション]「名前を変更」

このアクションはモデルフィールドノードの名前を変更します。モデルフィールドの名前を変更すると、そのフィールドにバインドされていたすべての可視コンポーネントのバインドが更新されます。

▼ [アクション]「削除」

このアクションはモデルオペレーションノードを削除します。モデルフィールドを削除すると、そのフィールドにバインドされていたすべての可視コンポーネントのバインドが解除されます。

モデルフィールドノードのプロパティ

あらゆるモデルフィールドノードが、以下で説明する複数の共通プロパティを共有します。また、モデルフィールドはそれぞれに、IDE が動的に検出し、提示する任意の個数の固有のプロパティがあり、それらプロパティはプロパティシートで設定できます。

▼ [プロパティ]「名前」

このプロパティは、モデルフィールドのインスタンス名を示します。IDE は生成するコードの一部としてこの名前のプロパティ値を使用し、ページの Java ソースファイル内でのモデルフィールドの作成およびアクセスを管理します。

モデルオペレーショングループノード

モデルオペレーショングループノードは、一群のモデルオペレーションノードの親ノードです。

モデルオペレーショングループノードのコンテキストメニューコマンド

▼ [アクション]「オペレーション...追加」

グループに新しいモデルオペレーションを追加します。モデルオペレーション用の新しいサブノードが追加されます。追加されたフィールドのデフォルト名は、モデルコンポーネントの作成者が定義したベース名にもとづく一意の名前です。開発者は、必要に応じて名前を変更できます。モデルオペレーションが追加されると、モデルの Java ファイルの生成コードの部分が更新されます。

▼ [アクション]「ペースト」

モデルオペレーションをペーストします。

▼ [アクション]「順序を変更」

このアクションは、現在のノードのサブノードの一覧をダイアログで表示します。このダイアログには、サブノードの表示順序を操作するためのボタン (「上へ移動」「下へ移動」) があります。

モデルオペレーショングループノードのプロパティ

▼ [なし]

モデルオペレーショングループノードのサブノード

モデルオペレーショングループノードには、モデルオペレーションサブノードが含まれます。

モデルオペレーションノード

モデルオペレーションノードでは、モデルオペレーションのプロパティおよび編集アクションに直接アクセスできます。

モデルオペレーションノードのコンテキストメニューコマンド

▼ [アクション]「カット」、「コピー」、「ペースト」

従来のカット、コピー、ペースト動作をするアクションです。モデルオペレーションを別のモデルに移動しても、プロパティにこのオペレーション名がある不可視コンポーネントが更新されることはありません。不可視コンポーネントでオペレーション名を手動で再設定するかどうかは、開発者が決定します。

▼ [アクション]「名前を変更」

このアクションはモデルオペレーションノードの名前を変更します。モデルオペレーションの名前を変更しても、プロパティにこのオペレーション名がある不可視コンポーネントが更新されることはありません。不可視コンポーネントでオペレーション名を手動で再設定するかどうかは、開発者が決定します。

▼ [アクション]「削除」

このアクションはモデルオペレーションノードを削除します。モデルオペレーションを削除しても、プロパティにこのオペレーション名がある不可視コンポーネントが更新されることはありません。不可視コンポーネントでオペレーション名を手動で再設定するかどうかは、開発者が決定します。

モデルオペレーションノードのプロパティ

あらゆるモデルオペレーションノードは、以下で説明する複数の共通プロパティを共有します。また、モデルオペレーションはそれぞれに、IDE が動的に検出し、提示する任意の個数の固有のプロパティがあり、それらプロパティはプロパティシートで設定できます。

▼ [プロパティ]「クラス」

このプロパティは、このモデルオペレーションの完全限定クラス名を示します。このプロパティは編集できません。

▼ [プロパティ]「名前」

このプロパティは、モデルオペレーションのインスタンス名を示します。IDE は生成するコードの一部としてこの名前のプロパティ値を使用し、ページの Java ソースファイル内でのモデルオペレーションの作成およびアクセスを管理します。

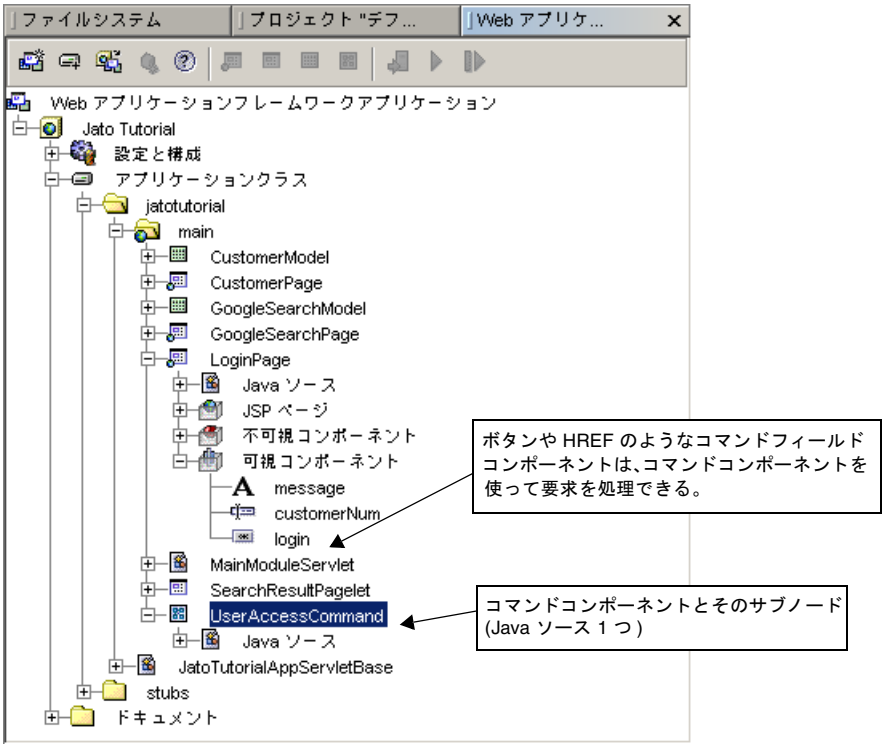
コマンドコンポーネントノード

コマンドは、アプリケーションのコントローラコンポーネントです。たとえばコマンドフィールド (ボタンまたは HREF) がクリックされたときに特定の種類のアクションを処理する目的で作成することができます。

コマンドコンポーネントは、アプリケーション全体の多数のさまざまなコマンドフィールドで再利用できます。また、コマンドコンポーネントを使用する多数のコマンドフィールドがコマンドコンポーネントに渡すパラメータに基づいて、その動作をさらにカスタマイズすることもできます。

コマンドコンポーネントの作成についての詳細は、『Web アプリケーションフレームワーク 開発ガイド』および『Web アプリケーションフレームワーク コンポーネント作成ガイド』をご覧ください。

次の図はコマンドコンポーネントノードの例を示しています。



コマンドコンポーネントノードのコンテキストメニューコマンド

▼ [アクション]「開く」

このアクションはソースエディタでコンポーネントの Java ソースファイルを開いて、そのクラスにジャンプします。

▼ [アクション]「イベント」

このアクションは、このコンポーネントへの実装に適したイベントハンドラメソッドの一覧からなるサブメニューを起動します。すでに実装されているイベントハンドラの名前は太字で表示されます。実装されていないイベントハンドラは太字で表示されません。

一覧から太字のイベントハンドラを選択すると、IDE がコンポーネントの Java ソースを開き、そのイベントハンドラの先頭にカーソルを移動します。太字でないイベントハンドラを選択すると、IDE がコンポーネントの Java ソースファイルに、対応するイベントハンドラのメソッドスタブを追加し、その上にカーソルを移動します。

利用可能なイベントハンドラメソッドの一覧はコンポーネントによって異なり、現在のコンポーネントのもとになっているコンポーネントによって宣言されています。コンポーネントによっては、イベントハンドラを宣言していないものもあります。コンポーネントがイベントハンドラを 1 つも宣言していない場合、このメニューコマンドは使用不可になります。

▼ [アクション]「コンポーネントメソッド」

このアクションは、このコンポーネントが実装可能な、有用なコンポーネントメソッドの一覧からなるサブメニューを起動します。すでに実装されているコンポーネントメソッドの名前は太字で表示されます。実装されていないメソッドの名前は太字で表示されません。

一覧から太字のコンポーネントメソッドを選択すると、IDE がコンポーネントの Java ソースを開き、そのメソッドの先頭にカーソルを移動します。太字でないコンポーネントメソッドを選択すると、IDE がコンポーネントの Java ソースファイルに対応するメソッドスタブを追加し、その上にカーソルを移動します。

利用可能なコンポーネントメソッドの一覧はコンポーネントによって異なり、現在のコンポーネントのもとになっているコンポーネントによって宣言されています。コンポーネントによっては、コンポーネントメソッドを宣言していないものもあります。

一般にコンポーネントメソッドは、利用可能なベースクラスメソッドのサブセットです。そのサブセットは、特にサブクラスによるオーバーライドを意図したもので、ライブラリコンポーネントの作成者がこの方法を使用して公開することに意味があるものです。コンポーネントがイベントハンドラを 1 つも宣言していない場合、このメニューコマンドは使用不可になります。

▼ [アクション]「名前を変更」

このアクションはコンポーネントクラス名を設定して、コンポーネントの Java ソースファイルの名前を変更します。

▼ [アクション]「削除」

このアクションはコンポーネントを削除します。コンポーネントの Java ソースファイルも削除します。

▼ [アクション]「カット」、「コピー」、「ペースト」

従来のカット、コピー、ペースト動作をするアクションです。

コマンドコンポーネントノードのプロパティ

▼ [プロパティ]「コンポーネントクラス」

このプロパティは、コンポーネントの完全限定クラス名を示します。このプロパティは編集できません。

▼ [プロパティ]「ComponentInfo クラス」

このプロパティは、コンポーネントの完全限定 ComponentInfo クラスを示します。このプロパティは編集できません。

▼ [プロパティ]「名前」

このプロパティは、コンポーネントクラス名を設定します。

コマンドコンポーネントのサブノード

コマンドノードには、「Java ソース」ノードというサブノードが 1 つだけあります。

「Java ソース」ノード

「Java ソース」ノードは、このコンポーネントの Java ソースファイルにノードレベルでアクセスできるようにします。Web アプリケーションフレームワークコンポーネントのノードは 「Java ソース」ノードの親になり、アプリケーションコンポーネントが論理的に単なる Java ノード 以上のものから構成されていることを強調します。以下の点を除けば、「Java ソース」ノードは、Sun Java Studio Enterprise 7 の従来の Java ソースコードのノードと同じです。

- Web アプリケーションフレームワークコンポーネントのノードが親である。
- ノード名は固定で、「Java ソース」である。
- コンテキストメニューのカット、削除、名前を変更コマンドは使用不可になる。これらのコマンドについては、コンテキストメニューではなく、コンポーネントのカット、削除、名前を変更コマンドを使用します。

「Java ソース」ノードのコンテキストメニューコマンドとプロパティについては、Sun Java Studio Enterprise 7 のマニュアルに完全な説明があります。

第 4 章

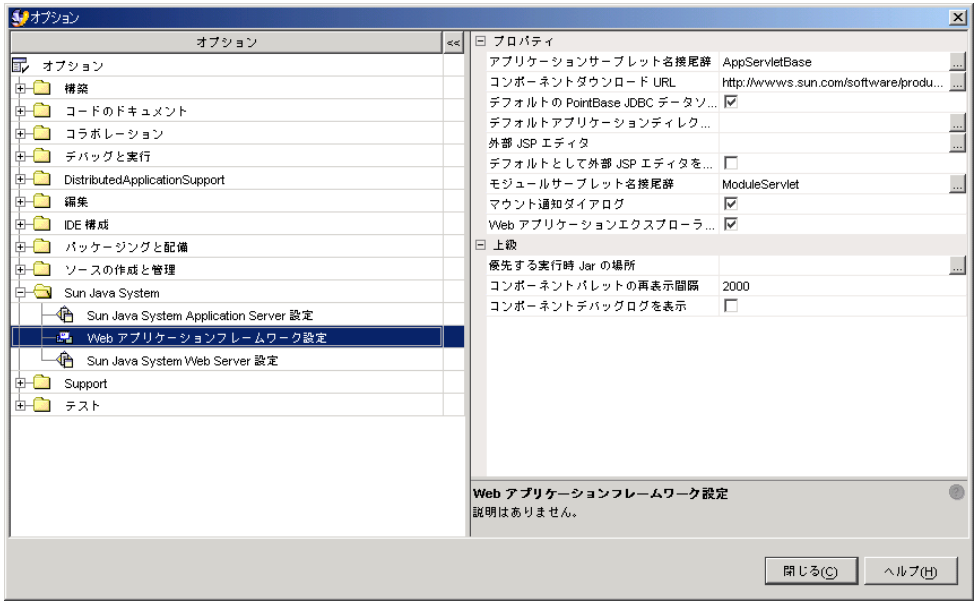
Sun Java Studio Enterprise 7 のツールオプション

この章では、Sun Java Studio Enterprise 7 のツールオプションの概要を説明します。

他の IDE 同様、Sun Java Studio Enterprise 7 には、IDE 全体と IDE の特定のモジュール (Web アプリケーションフレームワークなど) に適用される一般的な構成オプションがあります。ここでは、Web アプリケーションフレームワークモジュールに設定可能なプロパティに注目します。

IDE のメインメニューから「ツール」->「オプション」を選択してください。

次の図に示すような「オプション」ウィンドウが表示されます。



最上位のノードをすべて畳んだ (閉じた) 状態で「Sun Java System」ノードを見つけ、展開します。次に「Web アプリケーションフレームワーク設定」ノードを選択します。「オプション」ウィンドウの右部分にプロパティシートのタブ (「プロパティ」および「上級」) が表示されます。

「プロパティ」プロパティシート

「Web アプリケーションフレームワーク設定」プロパティシートの「プロパティ」ウィンドウには、以下のプロパティがあります。

▼ [プロパティ]「アプリケーションサーブレット名接尾辞」

新規アプリケーションウィザードを使った Web アプリケーションフレームワークアプリケーションの新規作成では、アプリケーションサーブレットクラスというサーブレットクラスの名前を指定するよう求められます。このクラスにデフォルトで付加される接尾辞は「AppServletBase」です。ただし、これは単なる習慣であり、必須ではありません。

このプロパティでは、新しい Web アプリケーションフレームワークアプリケーションを作成するときにあらゆるアプリケーションサーブレットクラス名に付加する接尾辞をカスタマイズすることができます。

▼ [プロパティ]「コンポーネントダウンロード URL」

「Web アプリケーションフレームワークアプリケーション」ウィンドウ内の「Web アプリケーションフレームワーク」ルートノードの「コンポーネントをダウンロード」コンテキストメニューオプションを選択したときに使用するデフォルト URL (`http://www.sun.com/sunone/`) を指定します。

この URL の目的は、開発者がダウンロードして Web アプリケーションフレームワークアプリケーションで利用可能な Sun 以外の Web アプリケーションフレームワークコンポーネントのリポジトリを含む Web サイトへのショートカットを提供することにあります。このマニュアルの作成時点では、そうした Web ベースのコンポーネントリポジトリは作成されていません。

▼ [プロパティ]「デフォルトの PointBase JDBC データソースを作成」

新規アプリケーションウィザードを使った Web アプリケーションフレームワークアプリケーションの新規作成では、デフォルトで `jdbc/jdbc-pointbase` という名前の JDBC データソースオブジェクトが自動的に作成されます。このデータソースは、Sun Java Studio Enterprise 7 ソフトウェアに付属しているサンプルの PointBase データベースにアクセスするよう事前に設定されています。詳しくは、このマニュアルの「JDBC データソース」ノードの説明を参照してください。

このプロパティを「False」に設定すると、新規アプリケーションウィザードがデフォルトの JDBC データソースを作成しなくなります。Web アプリケーションフレームワークアプリケーションにこの JDBC データソースが存在しないことの悪影響や実行時パフォーマンスの低下はありません。実際には JDBC データソースは設計時の作業で使用されるだけです。

▼ [プロパティ]「デフォルトアプリケーションディレクトリ」

新規アプリケーションウィザードを使った Web アプリケーションフレームワークアプリケーションの新規作成では、アプリケーションを作成する場所 (ディレクトリ) を指定するよう求められます。デフォルトの場所は Sun Java Studio Enterprise 7 のユーザーディレクトリです。このデフォルトの場所は、プラットフォーム (UNIX、Linux、Windows) によって異なり、Sun Java Studio Enterprise 7 をインストールしたときに設定されます。

このプロパティは、Web アプリケーションフレームワークアプリケーションの新規作成時に独自のデフォルトの場所を指定することを可能にします。このプロパティを設定することによって、新規アプリケーションの作成時に、アプリケーション開発用のディレクトリに移動する手間が省けます。

このプロパティはまた、Web アプリケーションフレームワークアプリケーションをマウントするときのデフォルトの場所にも使用されます。Web アプリケーションフレームワークアプリケーションの作成とマウントの詳細は、このマニュアルの第 1 章を参照してください。

▼ [プロパティ]「外部 JSP エディタ」

JSP のソースコードを直接編集することが難しく、Sun 以外の HTML/JSP WYSIWYG エディタを利用したい場合は、IDE のメインメニューから「ツール」->「オプション」を選択して、「Sun Java System」ノードを開き、「Web アプリケーションフレームワーク設定」を選択すると、Web アプリケーションフレームワーク設定の「外部 JSP エディタ」プロパティで、そのアプリケーションの実行可能ファイルのパスを指定することができます。

その後、Web アプリケーションフレームワークアプリケーションで JSP を右クリックして「外部エディタで編集」アクションを選択すると、外部エディタが起動し、その JSP を読み込むことができます。ファイルのロックや上書きの問題を回避するため、外部エディタと Sun Java Studio Enterprise 7 の両方で同時に JSP を開かないよう注意してください。

▼ [プロパティ]「モジュールサーブレット名接尾辞」

新規アプリケーションウィザードを使った Web アプリケーションフレームワークアプリケーションの新規作成、あるいはモジュールフォルダの新規作成では、モジュールサーブレットクラスというサーブレットクラスの名前を指定するよう求められま

す。詳しくは、このマニュアルの「モジュールフォルダ」ノードの説明を参照してください。このクラスにデフォルトで付加される接尾辞は `ModuleServlet` です。ただし、これは単なる習慣であり、必須ではありません。

このプロパティでは、新しいモジュールサーブレットクラスを作成するときにあらゆるモジュールサーブレットクラス名に付加する接尾辞をカスタマイズすることができます。

▼ [プロパティ]「マウント通知ダイアログ」

「True」に設定すると、Web アプリケーションフレームワークアプリケーションのファイルシステムがマウントされたときに、マウント通知ダイアログが表示されます。

▼ [プロパティ]「Web アプリケーションエクスプローラツールバーを表示」

「True」に設定すると、「エクスプローラ」ウィンドウ内の「Web アプリケーションフレームワークアプリケーション」ウィンドウの最上部に Web アプリケーションフレームワークツールバーが結合されます。

▼ [プロパティ]「デフォルトとして外部 JSP エディタを使用」

「外部 JSP エディタ」プロパティで外部 JSP エディタを指定した場合、このオプションを「True」に設定すると、デフォルトエディタとして、Sun Java Studio Enterprise 7 エディタではなく、外部エディタが起動されます。Web アプリケーションフレームワークアプリケーションのコンテキスト内で、jato タグの有無に関係なく JSP をダブルクリックすると、指定した外部エディタでその JSP が開かれます。

Sun Java Studio Enterprise 7 エディタで JSP を開くには、JSP を右クリックして、「開く」コンテキストメニューコマンドを選択します。Web アプリケーションフレームワークアプリケーション以外の Web アプリケーションの場合、デフォルトで Sun Java Studio Enterprise 7 エディタが開きます。このプロパティは Web アプリケーションフレームワークアプリケーション内の JSP に対してのみ影響します。

「上級」プロパティシート

「Web アプリケーションフレームワーク設定」プロパティシートの「上級」ウィンドウには、以下のプロパティがあります。

▼ [プロパティ]「コンポーネントパレットの再表示間隔」

Web アプリケーションフレームワークコンポーネントパレットの内容を再表示する間隔を制御します。Web アプリケーションフレームワークコンポーネントを含む新しいコンポーネントライブラリを追加する場合、あるいは Web アプリケーションフ

レームワークアプリケーションでページやページレット、モデルコンポーネントを新規作成すると、次の再表示でコンポーネントパレットにそのコンポーネントが現れます。

デフォルトよりも短い間隔でチェックされるように、この再表示レートを設定する必要はありません。たとえば 5 ミリ秒おきにコンポーネントパレットを再表示した場合、IDE に無用な負荷をかけることになります。

▼ [プロパティ]「優先する実行時 Jar の場所」

Sun Java Studio Enterprise 7 のツールモジュールに付属している Web アプリケーションフレームワークの JAR 実行時ファイル (`jato.jar`) の代わりに使用する JAR ファイルの場所です。

▼ [プロパティ]「コンポーネントデバッグログを表示」

「True」に設定すると、IDE 内のログファイルが表示されます。このファイルはコンポーネント開発者がコンポーネントまたはコンポーネントライブラリをデバッグするのに役立つことがあります。

索引

L

N

R

S

W

あ

い

う

お

か

け

こ

さ

し

す

せ

つ

て

と

ゆ

ら

る

ろ