

# Webアプリケーション開発者ガイド(基礎編)

リリース 1.0  
初版: 2003 年 8 月



*Muratec Information  
Systems.LTD.*

---

Webアプリケーション開発者ガイド(基礎編)、リリース 1.0

原本部品番号:W1B0101-01

原本名:Hayabusa Web Application Developer's Guide (Basic)、Release1

原本著者:長谷川 和彦

編集:久田 雅子

Copyright © 2002、MURATEC INFORMATION SYSTEMS, LTD. All rights reserved.

Printed in Japan

#### 制限付権利の説明

プログラム(ソフトウェアおよびドキュメントを含む)の使用、複製または開示は、ムラテック情報システムとの契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されています。

このドキュメントの情報は、予告なしに変更されることがあります。ムラテック情報システムは本ドキュメントの無謬性を保証しません。

\* ムラテック情報システムとは、ムラテック情報システム株式会社を指します。

#### 危険な用途への使用について

ムラテック情報システム製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。ムラテック情報システム社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、ムラテック情報システムおよびその関連会社は一切責任を負いかねます。

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

---

# 目次

<b>第Ⅰ部</b>	<b>サーブレット概要</b> .....	<b>1</b>
第1章	サーブレット概要.....	2
	1. サーブレットとは.....	2
	2. サーブレットコンテナ.....	2
	3. CGIとサーブレット.....	3
第2章	Web アプリケーション概要.....	4
	1. Web アプリケーションとは.....	4
	2. Web アプリケーションの構成.....	5
	3. Tomcat.....	6
<b>第Ⅱ部</b>	<b>セッション・トラッキング</b> .....	<b>7</b>
第3章	セッション・トラッキング.....	8
	1. セッション・トラッキング.....	8
第4章	サーブレットの作成.....	10
	1. HttpSession インターフェース.....	10
	2. セッション・オブジェクトの作成.....	10
	3. セッション・オブジェクトのメソッド.....	11
第5章	URL Rewriting.....	12
<b>第Ⅲ部</b>	<b>JSP</b> .....	<b>13</b>
第6章	JPS.....	14
	1. JSP とは.....	14
	2. JSP コンテナ.....	14
	3. JSP とサーブレット.....	15
	4. JSP ファイルの実行手順.....	16
	5. JSP の実行.....	19
	6. JSP サーブレットのライフサイクル.....	20
第7章	JSP の構成要素.....	21
	1. JSP の構成要素.....	21
	2. コメント.....	22
	3. スクリプトタグ.....	23
	4. ディレクティブタグ.....	24
	5. 定義済みオブジェクト.....	26
	6. アクションタグ.....	28
第8章	JavaBeans 概要.....	31

1. JavaBeans とは .....	31
2. JavaBeans の規則 .....	32
<b>第 IV 部 カスタムタグ .....</b>	<b>33</b>
第9章 カスタムタグ .....	34
1. カスタムタグとは .....	34
第10章 カスタムタグの作成手順 .....	36
1. カスタムタグの作成手順 .....	36
2. TLD ファイルの作成 .....	37

---

# はじめに

このマニュアルでは、Webアプリケーションの開発に関する機能について説明します。このマニュアルの内容は、すべてのプラットフォーム上で同様に動作する機能に適用されるもので、システム固有の情報は含みません。

## 《対象読者》

このマニュアルは、新しいアプリケーションを開発したり、既存のアプリケーションを実行できるように変換するプログラマを対象にしています。このマニュアルは、システム・アナリスト、プロジェクト管理者およびデータベース・アプリケーション開発に関心がある方にも役立ちます。前提として、MIS 認定 Web アプリケーション教育 入門コースを受講された方、あるいは同等レベルの知識を習得している方とします。

### 《本文の表記規則》

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則を示しています。

規則	意味
太字	太字は、本文中に定義されている用語または用語集に含まれている用語、あるいはその両方を示します。この句を指定する場合は、索引構成表を作成します。
大文字	大文字は、システムにより指定される要素を示します。
小文字	小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびサンプルのユーザー指定要素を示します。 <b>注意:</b> 一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。
イタリック	イタリックは、プレースフォルダまたは変数を示します。

### 《コード例の表記規則》

次の表は、コード例の記載上の表記規則を示しています。

規則	意味
[ ]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体は入力しないでください。
	縦線は、大カッコまたは中カッコ内の複数の選択肢を区切るために使用します。オプションのうち 1 つを入力します。縦線自体は入力しないでください。
… ⋮	省略記号は、例に直接関係のないコード部分が省略されていることを示します。

### 《アイコン》

本文中には、特別な情報を知らせるために、次のアイコンが用意されています。



#### ヒント

提案や秘訣を示し、これらによって、時間の節約や手順の容易化などを実現できる場合があります。



#### 警告

システムに致命的な影響を及ぼす可能性のあるアクションについて、注意が必要であることを示します。



#### コラム

関連する基礎知識や細かい技などを解説しています。

# 第 I 部

## サブレット概要

『義を見て為さざるは、勇なきなり。』  
孔子

(正しい行為と知りながら行動しないのは、  
勇気に欠けている証拠である。)

---

ここでは、サブレットと Web アプリケーションについて概要を説明します。  
構成は、次のとおりです。

第 1 章 サブレット概要  
サブレットの概要について説明します。

第 2 章 Web アプリケーション概要  
Web アプリケーションの概要について説明します。

Web  
Web  
アプリケーション

# 第1章 サブレット概要

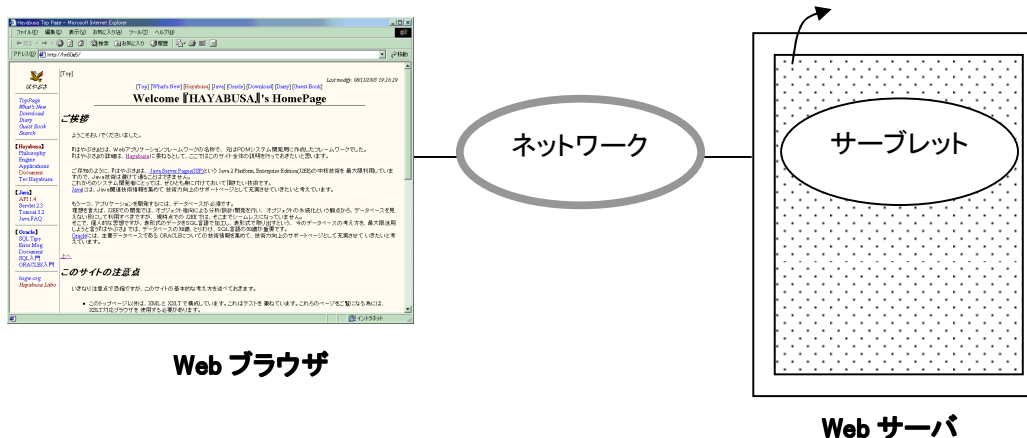
この章では、サブレットの概要について説明します。

## 1. サブレットとは

サブレットは、Web サーバ側で動作するJavaプログラムです。サブレットを使用することにより、サーバの機能を拡張することができます。アプレットはクライアント側のJavaプログラムであり、サブレットはサーバ側のJavaプログラムです。

## 2. サブレットコンテナ

サブレットコンテナとは、サブレットを実行するコンテナです。サブレットを実行するには、サーバ側にサブレットコンテナが必要です。



### ヒント

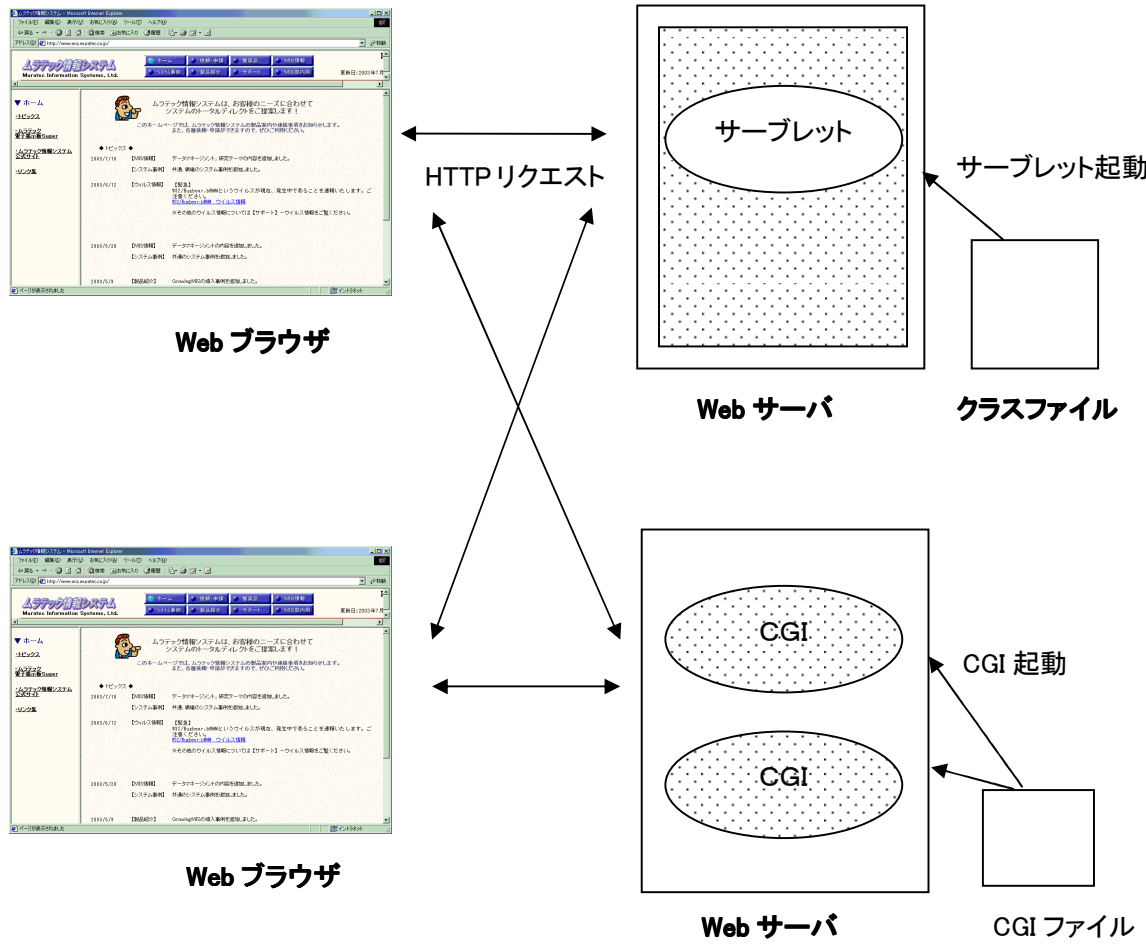
このコースでは、サブレット実行環境として、Tomcatを使用します。TomcatはApacheのjakartaプロジェクトによって作成されたフリーのサブレット実行環境です。jakartaプロジェクトについての詳細は、以下のURLを参照してください。

<http://Jakarta. Apache. Org/>



### 3. CGIとサーブレット

CGI(Common Gateway Interface)とは、サーバ側で動作するプログラムです。  
 Web サーバ側でCGIの設定をすることにより、Web ブラウザからの要求を Web サーバがCGIを起動して処理し、その結果をブラウザに返します。



**ヒント**

CGIとサーブレットはともに Web ブラウザからの HTTP リクエストに応答することができますが、それぞれに長所と短所があります。

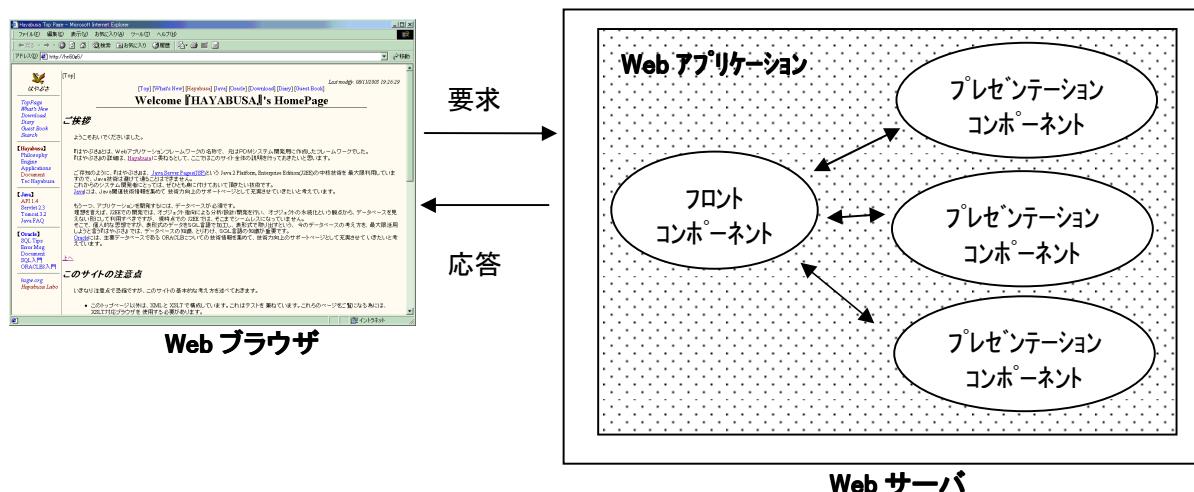
	CGI	サーブレット
言語	様々な言語でプログラムを作成できる プログラム言語が環境に依存する	CGIに比べて言語が特定される プログラム言語が環境に依存されない
負荷	Web サーバに大きな負担がかかる	CGIに比べて Web サーバへ負担がかからない
競合	他のhttpリクエストとの競合が発生しない(プロセス実行)	他のhttpリクエストとの競合する可能性がある(スレッドセーフではない)

# 第2章 Web アプリケーション概要

この章では、Web アプリケーションの概要について説明します。

## 1. Web アプリケーションとは

Web アプリケーションとは、サーブレットや JSP、HTMLファイルやイメージ・ファイルなどをまとめて一つのパッケージとして扱う概念です。Web アプリケーションには、フロント・コンポーネントとしての機能と、プレゼンテーション・コンポーネントとしての機能をもつことができます。フロントコンポーネントは、ユーザからの要求を処理したり、他のコンポーネントを管理するための機能を持ちます。このコンポーネントには、JSP やサーブレットなどを含めることができます。プレゼンテーション・コンポーネントは、ユーザに対する応答を生成する機能を持ちます。このコンポーネントには、JSP、HTML ファイルやイメージファイルなどを含めることができます。



### ヒント

Web アプリケーションは、プレゼンテーション・コンポーネントと、フロント・コンポーネントの機能を持つコンポーネントとしてとらえることができます。このコンポーネントのことを、Web コンポーネントと呼びます。



### ヒント

Web アプリケーションの構成を一つにまとめたファイルをWAR (WebApplicationArchive) ファイルと呼びます。



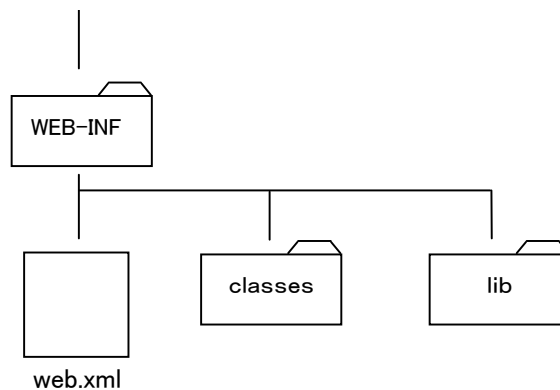
### ヒント

JSP については第Ⅲ部 JSP で詳しく説明します。

## 2. Web アプリケーションの構成

Web アプリケーションは、一つのディレクトリに必要なサーブレットなどをパッケージ化します。そのために Web アプリケーション用に作成されたディレクトリを、Web アプリケーションのルートディレクトリと考えることができます。

Web アプリケーションを構成するのに必要なものは、以下のとおりです。



### ①Web アプリケーションのルートディレクトリ

Web アプリケーション用ディレクトリを任意の名前で作成します。  
Tomcatには、Web アプリケーションを配置するディレクトリがあります。

### ②WEB-INFディレクトリ

Web アプリケーションのルートディレクトリに、WEB-INFディレクトリを作成します。

### ③WEB-INF/Web.xmlファイル

Web アプリケーションで使用する設定ファイルです。

### ④WEB-INF/classesディレクトリ

classesディレクトリは、Web アプリケーションで使用するクラスファイルを格納するディレクトリです。サーブレットはこのディレクトリにパッケージ化して配置します。

### ⑤WEB-INF/libディレクトリ

libディレクトリは、classesディレクトリと同様にクラスファイルを格納するディレクトリです。JARファイルを格納することができます。



#### ヒント

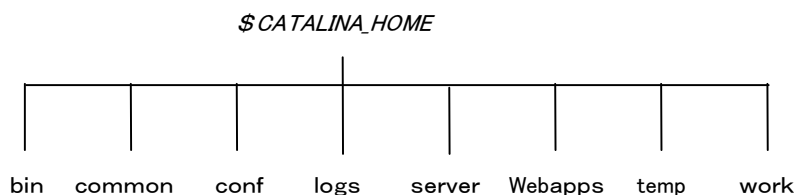
`$CATALINA_HOME`はTomcatのホームディレクトリパスになります。  
Web アプリケーションのルートディレクトリは、`$CATALINA_HOME/conf/server.xml` ファイルで設定しています。標準では、`G:\webapps` としています。

### 3. Tomcat

Tomcatはサーブレットを実行するためのサーブレット実行環境です。

#### ①Tomcatのディレクトリ構成

tomcatをインストールすると、以下のようなディレクトリ構成になります。



#### ②Tomcatの起動と停止

通常、Tomcatのサービスを起動するには、*\$CATALINA\_HOME* /bin/startup.bat を実行します。サービスを停止する場合は、*\$CATALINA\_HOME* /bin/shutdown.bat を実行します。  
Web エンジンの場合は、環境変数の設定を行う必要があるため、代りに、*%bin%startup.bat* と、*%bin%shutdown.bat* を実行します。

起動例

```
> %bin%startup.bat
```

停止例

```
> %bin%shutdown.bat
```



#### ヒント

Tomcat は、起動時に Web アプリケーションを認識します。新しく Web アプリケーションを作成した場合や、新しくサーブレットを作成した場合は、Tomcatのサービスを再起動する必要があります。



#### ヒント

変更のあった、Web アプリケーションを自動的に再ロードする機能などの管理機能は Web アプリケーション・サーバ製品により異なります。

Tomcat の場合は、*%CATALINA\_HOME%/conf/server.xml* で設定します。

## 第 II 部

## セッション・トラッキング

『学びて思わざれば則ちくらし、  
思いて学ばざれば則ちあやうし。』  
孔子

(他人からの知識や意見ばかり得ようとして、  
自分自身で考えないと本当の真実はわからない、  
逆に自分の考えばかりで他人の考えを理解  
しようとしなければ、独断におちいってしまう。)

---

ここでは、サーバとクライアントとの間でステートフルな環境を提供するセッション・トラッキングについて説明します。

構成は次のとおりです。

第 3 章 セッション・トラッキング  
セッショントラッキングの特徴について説明します。

第 4 章 サブレットの作成  
セッション・トラッキングを使用した、サブレットの作成について説明します。

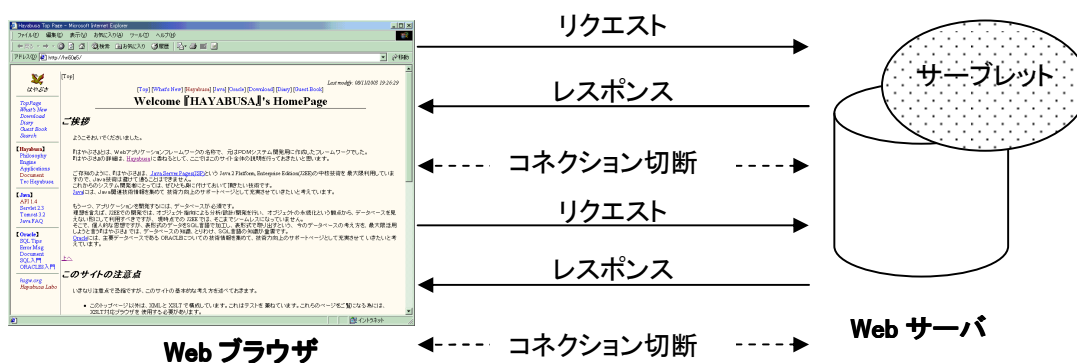
第 5 章 URL Rewriting  
クッキーが使用できない場合のセッショントラッキングの機能を利用する方法について説明します。

## 第3章 セッション・トラッキング

この章では、セッショントラッキングの特徴について説明します。

### 1. セッション・トラッキング

HTTP における処理はステートレスです。ステートレスとは、リクエストによってコネクションが確立し、レスポンスを返した後にコネクションが切断されるような処理のことです。つまり、このような仕組みではクライアントの情報を維持することができません。



セッション・トラッキングは Web 上でステートフルな状態を可能にするメカニズムです。サーブレットは、サーバ上に各クライアントごとのセッション・オブジェクトを用意することによって、クライアント情報を保持します。



#### ？ ヒント

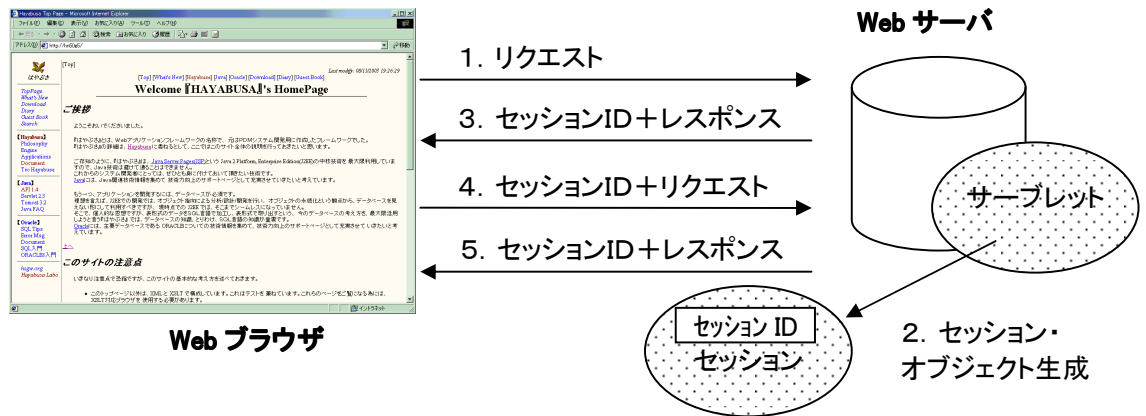
HTTP1.1 では、Web でステートフルな環境を提供するため、クッキーという文字列を使用することができます。

## ①セッション・オブジェクト

セッション・オブジェクトは、セッション・トラッキング機能を利用する際に、使用されるオブジェクトです。

クライアントから最初にサーブレットへリクエストをしたとき、サーブレットによってそのクライアントごとにセッション・オブジェクトが生成されます。この生成されたセッション・オブジェクトは Web サーバ上に保持されます。サーブレットはセッション・オブジェクトに様々な情報を追加したり、取り出したりすることができます。

また、セッション・オブジェクトごとに Web サーバから識別文字列が割り当てられます。この識別文字列をセッション ID と呼びます。セッション ID は、同じクライアントから再度リクエストが発生した時に、クライアントの識別をするために使用されます。



### ? ヒント

実際のセッション ID は、クッキーとして送られる文字列です。2回目以降のクライアントからのリクエストでは、このクッキーであるセッション ID をリクエスト・データとともに Web サーバに送り、Web サーバはこのセッション ID を、対象となるセッション・オブジェクトを検索するために使用します。

## ②セッションの持久性

クライアントからのアクセスが一定時間アイドル状態になると、クライアントのセッションは自動的に Time-Out となります。Time-Out 時にセッション・オブジェクトとそのオブジェクトが保持しているデータは消滅します。

セッション・オブジェクトは、サーブレットによって明示的に消滅させることもできますし、Time-Out 時間を変更することもできます。

### ? ヒント

Tomcat では、デフォルトで 30 分アイドル状態が続くと Time-Out となります。

## 第4章 サブレットの作成

---

この章では、セッション・トラッキングを使用した、サブレットの作成について説明します。

セッション・トラッキングを使用する場合には、`javax.servlet.http.HttpSession` インターフェースが提供されています。

### 1. HttpSession インターフェース

`HttpSession` インターフェースが提供する主なメソッドとして、`getAttribute()`メソッドと、`setAttribute()`メソッドがあります。

これらのメソッドは、機能的に、`Hashtable` クラスの機能に似ています。セッション・オブジェクトに保持されるデータはキーと値をペアにして扱います。つまり、セッション・オブジェクトは、様々な種類の情報を保持することができます。

### 2. セッション・オブジェクトの作成

セッション・トラッキングを利用したサブレットを作成するには、セッション・オブジェクトが必要です。セッション・オブジェクトを生成するには、`HttpServletRequest` クラスの `getSession()`メソッドを使用します。

- `HttpSession getSession(boolean create)`  
`create` に `true` を指定すると、クライアントからのリクエストに関連しているセッション・オブジェクトをリターンします。関連するオブジェクトがない場合は、新規に `HttpSession` クラスのオブジェクトを生成します。  
`create` に `false` を指定すると、クライアントからのリクエストに関連しているセッション・オブジェクトをリターンします。関連オブジェクトがない場合は、`null` をリターンします。



#### ヒント

セッション・オブジェクトのセッション ID をクライアントに送るためには、`getSession()`メソッドを使用して、セッション・オブジェクトを生成または獲得する必要があります。



#### ヒント

引数なし `getSession()`メソッドは、`getSession(true)`メソッドと同じ結果をリターンします。



### 3. セッション・オブジェクトのメソッド

セッション・オブジェクトを操作するには、HttpSession インターフェースで定義されているメソッドを使用します。

**①Object getAttribute(String name)**

name で指定した名前に相当するオブジェクトをリターンします。相当するオブジェクトがない場合は、null をリターンします。

**②void setAttribute(String name, Object value)**

name で指定した名前で、value に指定したデータをセッション・オブジェクトに設定します。

**③void removeAttribute(String name)**

name で指定した名前の値を削除します。

**④void setMaxInactiveInterval(int interval)**


セッション・オブジェクトの接続時間を設定します。クライアントからのアクセスが interval 秒なかった場合に、セッション・オブジェクトを消去します。

**⑤void invalidate()**

セッション・オブジェクトを明示的に消去します。

**⑥boolean isNew()**

セッション・オブジェクトが新規に生成されたものを判定します。

 **ヒント** setMaxInactiveInterval()メソッドに負数を設定すると、セッション・オブジェクトはタイムアウトしなくなります。

## 第5章 URL Rewriting

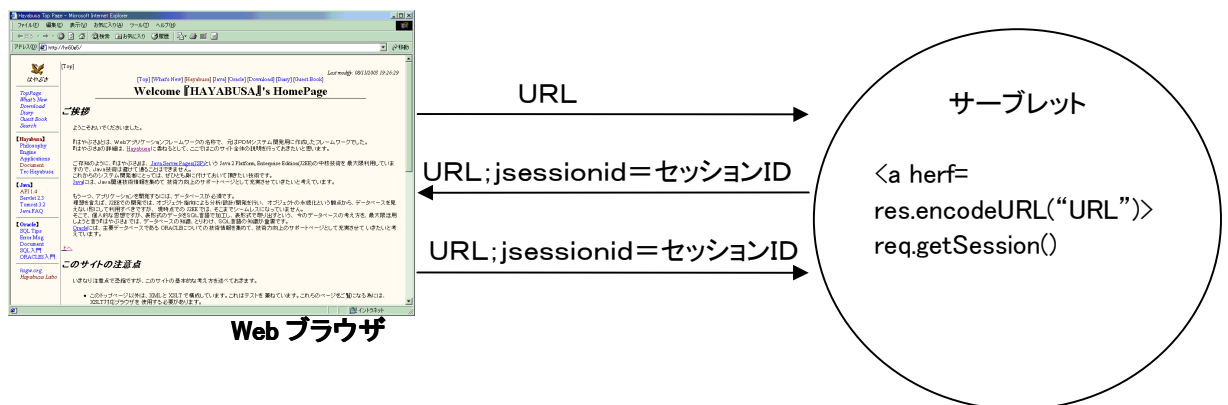
この章では、クッキーが使用できない場合のセッショントラッキングの機能を利用する方法について説明します。

セッション・トラッキングでは、クライアントと Web サーバのやりとりにおいて、セッション ID をクッキーとして扱いますが、環境によってはクッキーが使用できない場合もあります。このような状況においてもセッション・トラッキングの機能を利用するには、URL Rewriting を使用します。URLRewriting は、クライアントに送信する URL に対してセッション ID を付加することで、クライアントとセッションを結びつける機能です。クライアント側がアンカーをクリックすると、セッション ID を含んだ URL がサーバに送られます。Web サーバは送られてきた URL からセッション ID を取り出し、対象となるセッション・オブジェクトを見つけます。

### ①URL にセッション ID を付加するには

URL Rewriting は、URL のリンクに対してセッションIDを付加します。そのときに使用されるメソッドは、HttpServletResponse クラスの `encodeURL()` メソッドになります。

- String `encodeURL(String url)`  
URL にセッション ID を付加する。



#### ヒント

「URL;jsessionid=セッション ID」の形式は Servlet2.3 の仕様で決められています。ただし、セッション ID の生成ロジックに関しては環境に依存します。

# 第 III 部 JSP

『君子は義にさとる、小人は利にさとる。』

孔子

立派な人は正しさを優先し、凡人は自分の利益を優先させる。

---

ここでは、JSP (JavaServer Pages) の特徴、JSP ファイルの実行手順、JSP の構成要素について説明します。

構成は次のとおりです。

## 第 6 章 JSP 概要

JSP の特徴と実行手順について説明します。

## 第 7 章 JSP の構成要素

JSP の各構成要素について説明します。

## 第 8 章 JavaBeans 概要

JavaBeans の概要について説明します。

## 第6章 JPS

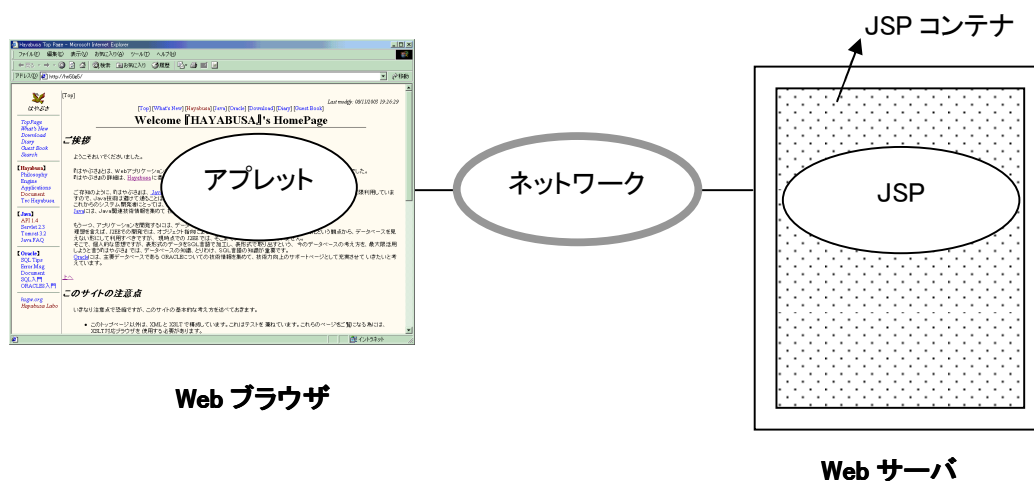
この章では、JSP (JavaServer Pages) の特徴と JSP ファイルの実行手順について説明します。

### 1. JSP とは

JSP (JavaServer Pages) は、HTML ファイルに Java プログラムを埋め込むことにより、Web サーバ側で動的な Web ページを生成する技術です。JSP を使用することにより、HTML タグに似た、JSP 独自のタグを用いサーバの機能を拡張することができます。

### 2. JSP コンテナ

JSP コンテナとは、JSP を実行するための実行環境です。JSP を実行するには、サーバ側に実行環境が必要です。



#### ヒント

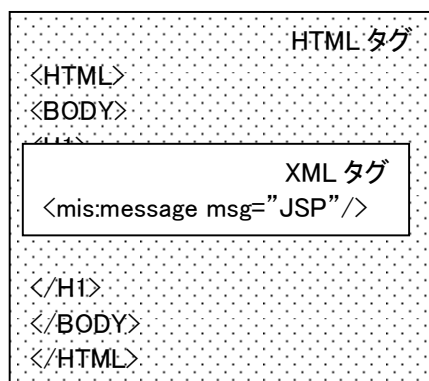
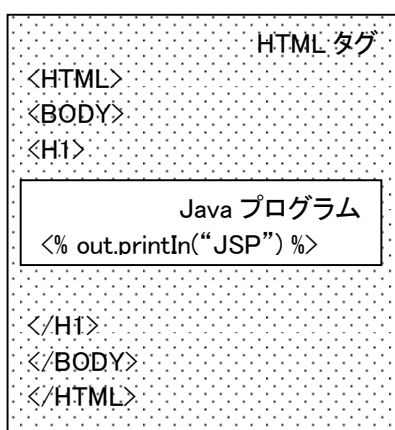
Tomcat は JSP をサポートしています。

### 3. JSP とサーブレット

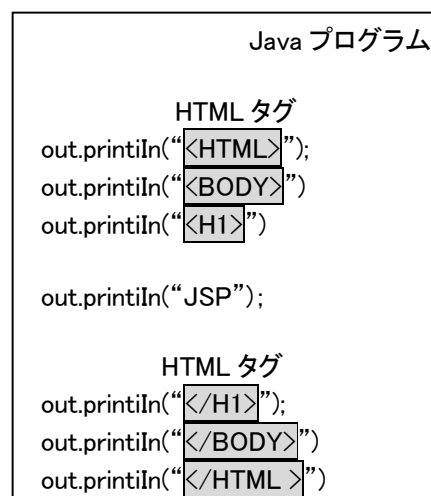
JSP では、HTML ファイルに JSP 独自の構成要素を用い Java プログラムを埋め込むことができます。サーブレットでは、Java プログラムに HTML のタグを埋め込み作成します。JSP とサーブレットはどちらも動的な Web ページを生成することができます。

#### JSP とサーブレットのコード比較

##### JSP ファイル(一般)



##### サーブレット



#### ? ヒント

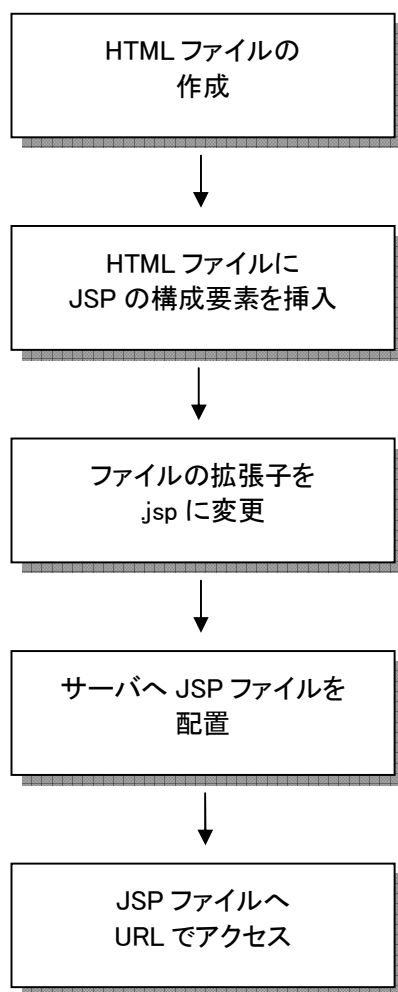
JSP ファイルの拡張子は、.jsp にします。  
JSP を使用する場合、javac によるコンパイルは必要ありません。

#### ? ヒント

エンジンで JSP を使用する場合、スクリプトレットを使用するのではなく、カスタムタグを使用します。

#### 4. JSP ファイルの実行手順

JSP ファイルの一般的な実行手順は、以下のとおりです。



## ①HTML ファイルの作成

HTML ファイルの作成は、Web デザイナーによって作成されたり、オーサリングツールなどを使用して作成します。

helloworld. html

```
<HTML>
:
<BODY>
:
<!--動的コンテンツが入る -->
:
</BODY>
</HTML>
```

## ②HTML ファイルに JSP の構成要素を挿入

作成された HTML ファイルに、JSP の構成要素を挿入します。

helloworld. html

```
<HTML>
:
<BODY>
:
<FONT size="20"><% out.print(" JSP Write Line.");%></FONT>
:
</BODY>
</HTML>
```

### ③HTML ファイルの拡張子を.jsp に変更

拡張子を.jsp へ変更することで、JSP ファイルとして処理します。

helloworld.jsp

```
1 <HTML>
2 <HEAD>
3   <TITLE>Hello JSP World<TITLE>
4 </HEAD>
5 <BODY>
6   <H1>Hello JSP World</H1>
7   <BR><HR><BR>
8
9   <P align="center">
10     <FONT size="20"><% out.print(" JSP Write Line.");%></FONT>
11   </P>
12
13 </BODY>
14 </HTML>
```

### ④サーバへ JSP ファイルを配置する

JSP ファイルは、Web アプリケーションのルート以下の jsp ディレクトリ以下へ配置します。

%UAP%/webapps/システムID/jsp/ 以下のディレクトリへ配置します。

### ⑤JSP ファイルへ URL でアクセスする

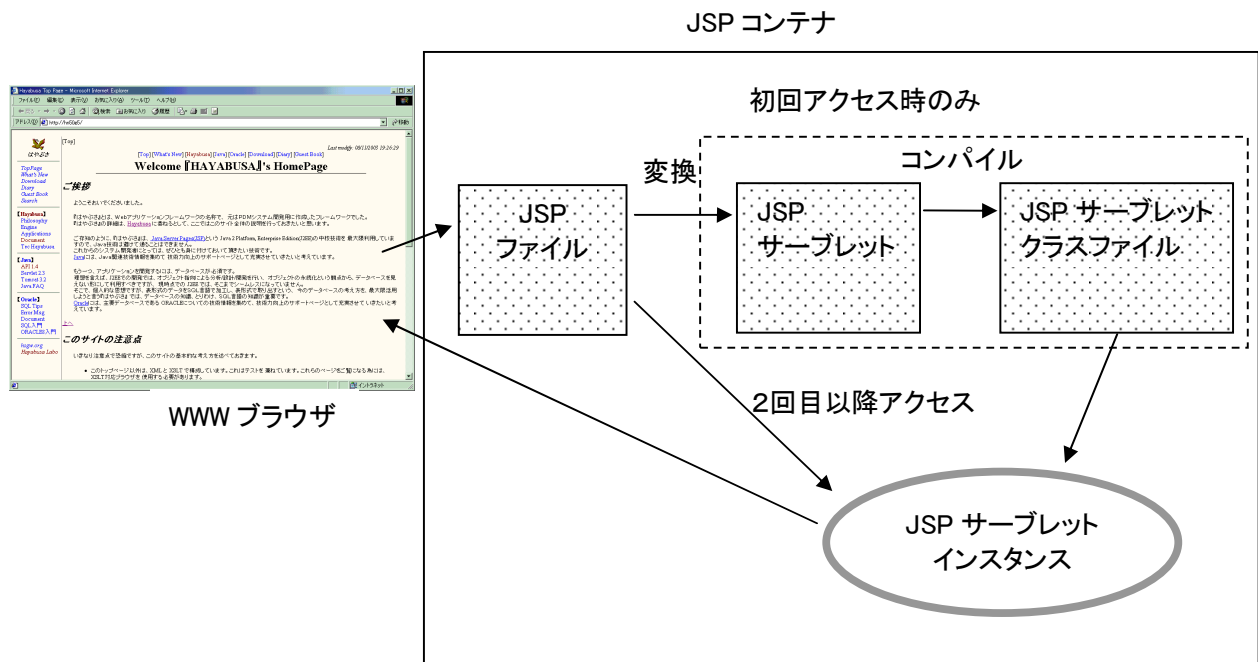
ブラウザから、以下の URL にアクセス。

http : //サーバ名 : 8080/システム I D/jsp /helloworld.jsp



## 5. JSP の実行

Web ブラウザから JSP ファイルにアクセスしたとき、JSP コンテナにより JSP サーブレットのソースコードに変換され、その後、コンパイルをして JSP サーブレットのクラスファイルが生成されます。実際にクライアントからのアクセスを受けるときには、JSP サーブレットのインスタンスとしてリクエストを処理します。

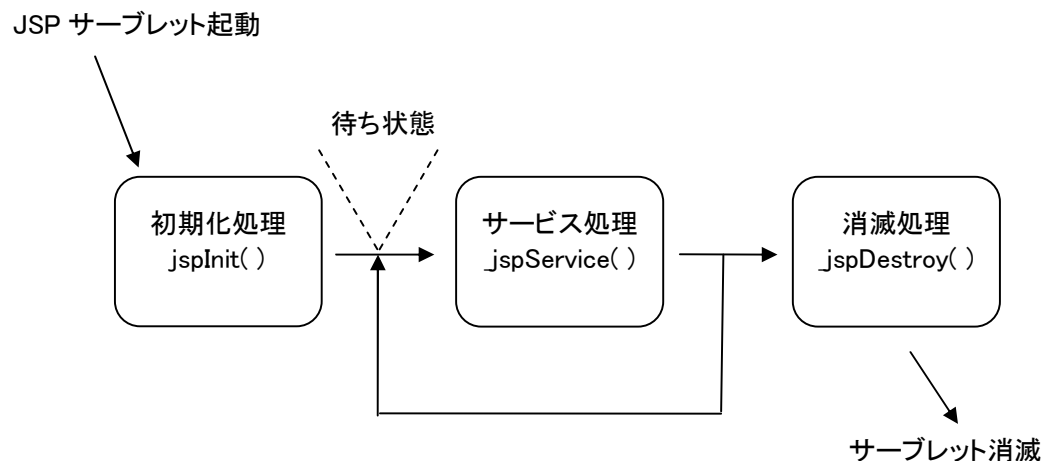


### ヒント

JSP サーブレットは、サーブレットと同様にインスタンスとして動作しますが、JSP ファイル作成者は、JSP サーブレットを意識する必要はありません。

## 6. JSP サーブレットのライフサイクル

サーブレットと同様に JSP サーブレットにもライフサイクルがあり、この流れに沿ってプログラムが動作します。JSP のライフサイクルは、`javax.servlet.jsp.JspPage` インターフェイスで提供されているメソッドと密接な関係があります。ただし、一般的にサーブレットと違い、JSP ファイル作成者が直接 `javax.servlet.jsp.JspPage` インターフェイスのメソッドを使用しません。



JSP は、最初にアクセスされたとき、JSP コンテナにより JSP サーブレットのソースコードに変換され、クラスファイルへコンパイルされ起動します。JSP サーブレットは起動されたとき、1 度だけ `jspInit()` メソッドにより初期化されます。クライアントが JSP ファイルに対してリクエストすると、`jspService()` メソッドによって応答します。この `jspService()` メソッドはマルチスレッドで動作します。

一度実行された JSP サーブレットは、クライアントからのリクエストに応じた後、そのまま常駐し、クライアントからの次のリクエストを待ちます。2 回目以降のリクエストは、待機している JSP サーブレットが応答します。また、続いて他のクライアントからリクエストがあると、常駐しているサーブレットがそのリクエストに応答します。これにより、その JPS サーブレットの変換、コンパイル、初期化処理が初回のみになるため、効率よく動作します。

ただし、スレッド・セーフなモデルではないため、変数などを共有する場合は、サーブレットと同様に `synchronized` で対処する必要があります。

## 第7章 JSP の構成要素

この章では、JSP の各構成要素について説明します。

### 1. JSP の構成要素

JSP ファイルは、HTML ファイルをベースに JSP 独自の構成要素を埋め込み作成します。JSP 独自の構成要素は、以下のとおりです。

#### コメント

JSP ファイルの中に2種類のコメントを記述することができます。

#### スクリプトタグ

スクリプトタグを使用することにより、JSP ファイルの中に Java 言語仕様に従った、Java プログラムを記述することができます。

#### ディレクティブタグ

ディレクティブタグを使用することにより、JSP の実行環境である JSP コンテナに対して指示を出すことができます。ディレクティブタグは JSP ファイルが変換されるとき JSP サーブレットに埋め込まれます。

#### 定義済みオブジェクト

定義済みオブジェクトを使用することにより、サーブレットと同様のオブジェクトを JSP ファイル中で宣言せずに使用することができます。

#### アクションタグ

アクションタグを使用することにより、オブジェクトの生成など特定の動作を行うことができます。



#### 警告

上記構成要素は、JSP 一般としての説明です。  
Web エンジンでは、JSP はXMLファイルとして記述しています。  
ディレクティブタグを除くすべてのタグは、原則使用禁止です。  
これは、JSP でのロジックを複雑にしないためです。

## 2. コメント

Web ブラウザに対し出力する HTML ファイルのコメントを入れることができます。コメントには出力コメントと隠れたコメントの2種類を使用することができます。

### ①出力コメント

通常の HTML や XML のコメントで、クライアントに出力するコメントを記述します。このコメントは Web ブラウザでソースを表示することにより確認可能です。また、Java の式を記述することができ、Web ブラウザによりページがロードされるときに処理されます。

```
<!-- コメント [<%=式%>] -->
```

### ②隠れたコメント

JSP 独自のコメントで、クライアントに出力しないコメントを記述します。Web ブラウザで表示させたくないもの、スクリプトタグや HTML タグのコメントアウトなどに使用します。

```
<%-- コメント--%>
```



#### ヒント

<%--コメント--%>の記述は<%/＊コメント\*/%>と同じ定義になります。

### 3. スクリプトタグ

HTML ファイルに基本的なスクリプトタグを埋め込み、ファイルの拡張子を.jsp にすることにより、動的な Web ページを作成することができます。

#### ①宣言

JSP ファイル内のスクリプトで有効な変数やメソッドの宣言を記述します。インスタンス変数、static 変数、メソッドを宣言できます。セミコロンを区切りとして、複数の宣言をすることが可能です。構文は Java 言語仕様に基づいたものでなければなりません。

```
<%! 宣言 %>
```

- 宣言はセミコロンで終了します。
- JSP ファイル内で使用する前に変数、メソッドの宣言ができます。
- page ディレクティブタグでインポートしたパッケージで宣言している変数、メソッドは宣言なしで使用することができ、JSP ページ内のスクリプトで有効な Java の式を記述できます。

#### ②式

JSP ファイル内のスクリプトで有効な Java の式を記述します。Java の式は String として解釈され、HTML 生成時に動的に処理されます。

```
<% Java の式 %>
```

- 式はセミコロンで終了できません。
- Java 言語仕様に従った構文そのまま記述できます。

#### ③スクリプトレット

JSP ページ内のスクリプトで有効な Java のコードを記述します。

```
<% Java のコード %>
```

- 定義済みオブジェクトや<jsp:useBean>タグで宣言したオブジェクトを使用することができます。
- Java 言語仕様に従った構文をそのまま記述できます。



#### ヒント

スクリプトタグやディレクティブなどは JSP ドキュメント内に XML 形式のタグとして、記述することもできます。例として、スクリプトレットは、<jsp:scriptlet>というタグで記述することができます。



#### 警告

エンジンでは、スクリプトタグの使用は、許可されていません。  
カスタムタグを使用してください。

## 4. ディレクティブタグ

ディレクティブタグは、JSP コンテナに対し、JSP ファイルを出力するときの指示をします。ディレクティブタグの属性で指定した指示は、JSP ファイル変換時に JSP サーブレットのソースコードに埋め込まれます。

### ①page ディレクティブタグ

page ディレクティブタグは JSP ファイル全体に関する属性を定義します。

```
<%@page[language="java"]
    [extends="パッケージ名.クラス名"]
    [import="インポートするパッケージ"]
    [session="true | false"]
    [contentType="MIME タイプ [;charset=キャラクタセット]"]
    [buffer="none | sizekb"]
    [autoFlush="true | false"]
    [info="info_text"]
%>
```

language

スクリプトレット、式、宣言で使用するスクリプト言語を指定します。現在使用できるキーワードは java のみです。

extends

JSP ファイルを JSP サーブレットに変換するときに JSP サーブレットが継承するクラスを指定します。Java 言語仕様の extends に相当します。

import

スクリプトレット、式、宣言で使用するパッケージをインポートします。java 言語仕様の import に相当します。JSP ファイル変換時、デフォルトでインポートされるパッケージは以下のとおりです。

- java.lang
- javax.servlet
- javax.servlet.http
- javax.servlet.jsp

### ? ヒント

Web サーバによっては、上記以外のパッケージがデフォルトでインポートされている場合もあります。ただし、ディレクティブの設定は language 属性が"java"の時のみに有効です。

**session**

JSP ファイルでセッション・トラッキングを使用するか否かを指定します。true ならばセッション・トラッキングを使用します。false ならば使用しません。デフォルトは true です。

**contentType**

JSP ファイルのレスポンスの MIME タイプ、エンコーディング方式を指定します。デフォルトは MIME タイプが text/html でエンコーディング方式が、ISO-8859-1 です。

**buffer**

クライアントへの出力ストリームのバッファサイズを指定します。none ならば使用しません。

**autoFlush**

バッファリングされたデータを自動的にフラッシュするか否かを指定します。デフォルトは true です。

**info**

Servlet インタフェースの `getServletInfo()` メソッドで返される文字列を指定します。

**ヒント**

page ディレクティブの属性はここで説明したもの以外にもあります。

### ②include ディレクティブタグ

include ディレクティブタグは、JSP ファイルに静的な HTML ファイルなどのデータを埋め込むときに使用します。JSP ファイルが JSP サーブレットに変換されるときに埋め込まれます。

```
<%@ include file="relativeURLspac" %>
```

### ③taglib ディレクティブタグ

taglib ディレクティブタグは JSP ファイル内でカスタムタグを利用するときに使用します。uri 属性で指定されたタグライブラリ URL と prefix 属性で指定されたプレフィックスで利用します。

```
<%@ taglib uri="tagLibraryURI" prefix="tagPrefix" %>
```



#### ヒント

taglib ディレクティブについての詳細は、第 V 部 カスタムタグで詳しく説明します。

## 5. 定義済みオブジェクト

JSP ではいくつかの定義済みオブジェクトが提供されています。定義済みオブジェクトはスクリプトレットと式で、宣言せずに使用することができます。

### ①request オブジェクト

Web クライアントからのリクエストの情報をカプセル化したオブジェクトを表します。

タイプ : javax.servlet.HttpServletRequest  
スコープ : request  
メソッド : getParameter(), getParameterNames()  
getParameterValues()

### ②response オブジェクト

Web クライアントへ送信する情報をカプセル化したオブジェクトを表します。

タイプ : javax.servlet.HttpServletResponsse  
スコープ : page  
メソッド : JSP のページの作成者は一般的に使用しません。

### ③session オブジェクト

セッション・トラッキングで使用する HttpSession オブジェクトを表します。

タイプ : javax.servlet.HttpServletRequestSession  
スコープ : session  
メソッド : isNew(), getAttribute(), setAttribute()



**④out オブジェクト**

Web クライアントへの出力ストリームを表します。

タイプ : javax.servlet.jsp.JspWriter  
スコープ : Page  
メソッド : println(),close(),flush()

**⑤config オブジェクト**

JSP サーブレットの ServletConfig オブジェクトを表します。

タイプ : javax.servlet. ServletConfig  
スコープ : Page  
メソッド : getServletContext(),getServletName(),

**⑥pageContext オブジェクト**

JSP サーブレットの PageContext オブジェクトを表します。

タイプ : javax.servlet.jsp. pageContext  
スコープ : Page  
メソッド : getServletConfig(),getRequest(),getResponse()

**⑦application**

JSP サーブレットの ServletContext オブジェクトを表します。

タイプ : javax.servlet. ServletContext  
スコープ : application  
メソッド : getAttribute(),setAttribute (),getContext()

## 6. アクションタグ

アクションタグは特定の動作を実行するタグです。JSP で標準定義されているアクションタグでは、オブジェクトの生成、プロパティの変更やページのインクルード、フォワードなどがあります。

### ①<jsp:useBean>タグ

<jsp:useBean>タグは、指定されたスコープにある JavaBeans のインスタンスを獲得します。

```
<jsp:useBean id="参照名">
    [scope=" page | request | session | application"]
    typeSpec />
```

id

JavaBeans インスタンスの参照名を指定します。すでに JavaBeans インスタンスが存在するときその参照をリターンし、存在しなければ、新たに JavaBeans インスタンスを生成し、参照をリターンします。

scope

JavaBeans インスタンスへの参照が利用できるスコープを指定します。デフォルトは page です。

- page : 現在の JSP ページを処理している間で参照できます。
- request : 一つのリクエストを処理している間で参照できます。
- session : 一つのセッションを処理している間で参照できます。
- application: Web アプリケーション全体で参照できます。

typeSpec

typeSpec で指定できるものは以下のとおりです。

- class="クラス名"
- class="クラス名" type="タイプ名"
- beanName="bean 名" type="タイプ名"
- type="タイプ名"

class

参照する JavaBeans のクラス名を指定します。

type

参照する JavaBeans のタイプ名を指定します。

beanName

参照する JavaBeans の Bean 名を指定します。

## ②<jsp:setProperty>タグ

<jsp:setProperty>タグは、<jsp:useBean>タグで獲得した JavaBeans のインスタンスのプロパティの値を設定します。

```
<jsp:setProperty name="参照名">
    property="プロパティ名"
    [[value=" 値" ] | [param="パラメータ名"]]
/>
```

name

<jsp:useBean>タグの id 属性で指定した名前を指定します。

property

値を設定したい JavaBeans インスタンスのプロパティ名を指定します。

value

property で指定した JavaBeans インスタンスのプロパティに設定する値を指定します。

param

HTTP リクエストで from から送られてきたパラメータを指定します。

## ③<jsp:getProperty>タグ

<jsp:getProperty>タグは、<jsp:useBean>タグで生成した JavaBeans のインスタンスのプロパティの値を獲得します。

```
<jsp:getProperty name="参照名" property="プロパティ名"/>
```

## ④<jsp:forward>タグ

<jsp:forward>タグは現在のリクエストを他の JSP やサーブレットに転送します。

```
<jsp:forward page="relativeURLspac"/>
```

## ⑤<jsp:include>タグ

<jsp:include>タグは JSP ファイルなどの他のファイルを動的にインクルードします。

```
<jsp:include page="relativeURLspac" [flush="ture | false"] />
```

### ⑥<jsp:param>タグ

<jsp:include>タグまたは、<jsp:forward>タグで指定した URL に対してパラメータを設定します。

```
<jsp: include page="URL" flush="ture">
    <jsp:param name ="paramNmae1" value="paramValue1" />
    <jsp:param name ="paramNmae2" value="paramValue2" />
    :
    :
</jsp:include>
```

## 第8章 JavaBeans 概要

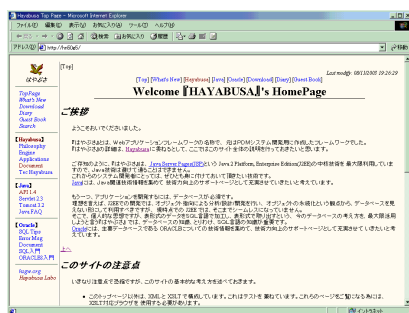
この章では、JavaBeans の概要について説明します。

### 1. JavaBeans とは

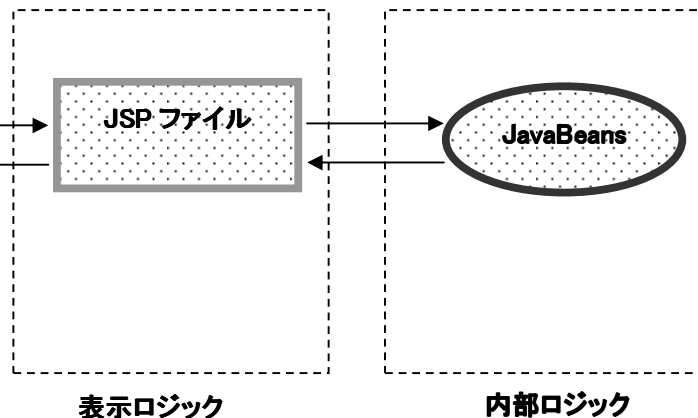
JavaBeans は移植性のある、プラットフォームに依存しない Java 言語で書かれたコンポーネントモデルです。

JavaBeans の開発者はこのアーキテクチャーを使うことによって、再利用可能なコンポーネントを Java 言語で作成することができます。

JavaBeans の各ソフトウェア部品 (コンポーネント) を Bean と呼びます。



Web ブラウザ



### 2. JavaBeans の規則

Java のクラスを Bean とするためには、いくつかの規則があります。

- public 指定のクラスとする。
- public 指定の引数なしのコンストラクタ定義する。
- パッケージ化する。  
Bean は、コンポーネントとしてパッケージ化することが推奨されます。
- クラスをシリアライズ可能とする。  
クラスがシリアライズ可能であることで、Bean の状態を保持することが可能になります。

クラス定義の例

```
package beans;
public class StringBean implements Serializable{
    :
    public StringBean(){
        :
    }
    :
}
```

- setter メソッド/getter メソッドを用意する  
クラスの属性値にアクセスするためのメソッドを定義する。

setter メソッド/getter メソッドの例

```
private int width;
:
public void setWidth(int width){
    this.width = width;
}
public int getWidth(){
    return this width;
}
```



#### ヒント

一般的に、変数に対するsetter メソッド/getter メソッドの定義は Java 言語の命名規則に従います。

例:

変数名:accessCounter → getter メソッド:getAccessCounter()

## 第 IV 部 カスタムタグ

『徳は孤ならず、かならず隣（となり）あり。』  
孔子

(立派な考えは、孤立しない。必ず協力者がいるものだ。)

---

ここでは、カスタムタグの特徴、カスタムタグの作成手順について説明します。  
構成は次のとおりです。

第 9 章 カスタムタグ概要  
カスタムタグの概要について説明します。

第 10 章  
カスタムタグの作成手順について説明します。

Web  
Web  
アプリケーション

---

## 第9章 カスタムタグ

---

この章では、カスタムタグの概要について説明します。

### 1. カスタムタグとは

カスタムタグは実行時にページ上で何らかの動的アクションを起こしたり、動的なコンテンツを生成するための XML タグです。カスタムタグは JSP ページ内のスクリプト要素と置き換えるために使用されます。

カスタムタグはプログラムを表示部分から切り離すときに役立ちます。このため、Web デザイナーがプログラムロジックへの影響を受けずにレイアウトを変更することができるようになります。

また、カスタムタグ開発者は前に作成したカスタムタグを再利用して、別の Web アプリケーションでも利用することができるようになります。

#### ①タグライブラリ

タグライブラリとはカスタムタグをまとめたものです。タグライブラリは独自に作成するものもありますが、種類としては 4 種類あります。タグライブラリの種類は以下のとおりです。

- サードパーティが提供するタグライブラリ  
サードパーティが提供する各 Web アプリケーションサーバに共通または特化したタグライブラリ。オープンソースの場合もあり、主に JakartaProject の taglibs などがある。
- Web アプリケーション・サーバ・ベンダが提供するタグライブラリ  
各 Web アプリケーション・サーバ上で特化した機能を提供するタグライブラリ。他ベンダの Web アプリケーション・サーバ上ではサポートしない場合がある。
- JSTL  
どの Web アプリケーション・サーバでも共通して使用できるタグライブラリ。JSTL の標準化は JCP の JSR-052 において策定され、JakartaProject によってリファレンス実装が行われている。
- 独自に開発するタグライブラリ  
独自に作成するタグライブラリ。Web アプリケーションサーバ・ベンダで提供され独自に作成されていない機能を持つカスタムタグを作成することができる。

本章では独自に開発するタグライブラリを紹介します。

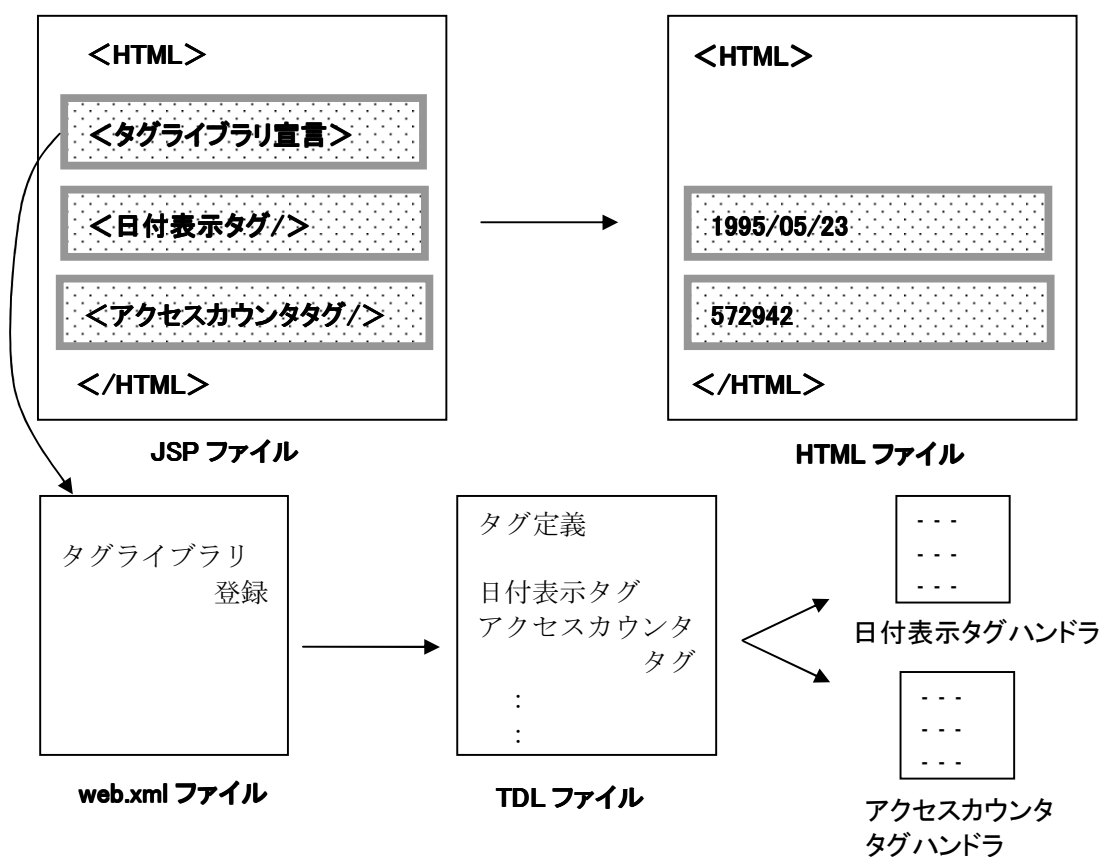


## ②カスタムタグライブラリの関連ファイル

カスタムタグを作成するにはいくつかの関連するファイルが必要です。必要なファイルは以下のとおりです。

- TLD ファイル  
タグライブラリを定義するための XML ファイル。
- タグハンドラ・クラスファイル  
カスタムタグに関連付けられたクラスファイル。
- web.xml ファイル  
Web アプリケーションにタグライブラリを登録するための XML ファイル。
- JSP ファイル  
カスタムタグを利用する JSP ファイル

### カスタムタグの関連



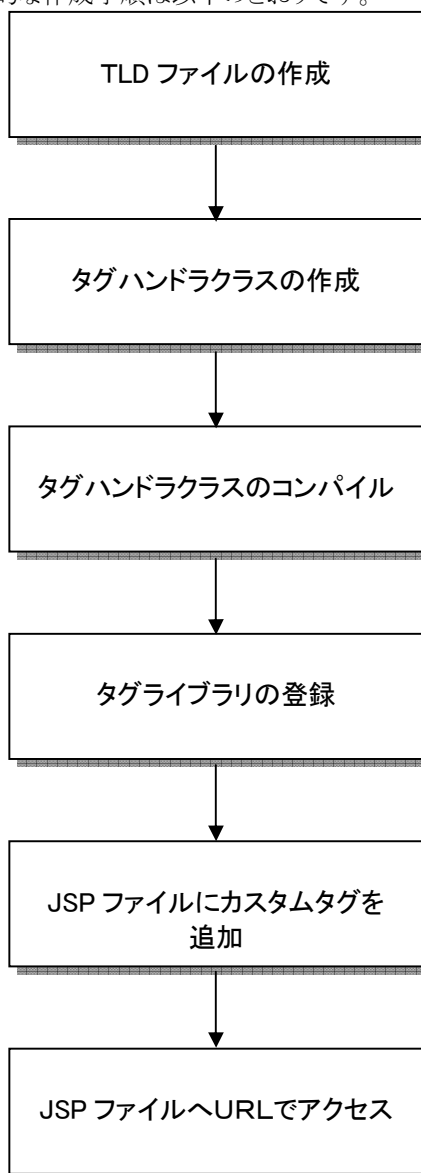
## 第10章 カスタムタグの作成手順

---

この章では、カスタムタグの作成手順について説明します。

### 1. カスタムタグの作成手順

カスタムタグの一般的な作成手順は以下のとおりです。



## 2. TLD ファイルの作成

TLD ファイルは、JSP ファイルから利用されるタグライブラリを定義する XML ファイルです。この中には複数のカスタムタグを定義することができます。  
作成するファイルの拡張子は.tldです。

### ①<taglib>タグ

TLD ファイルは<taglib>タグによって定義します。<taglib>タグ内の主な構成要素は以下のとおりです。

- <tlib-version>  
タグライブラリのバージョン
- <jsp-version>  
JSP のバージョン
- <short-name>  
JSP オーサリングツールなどで使用するタグライブラリの名前
- <tag>  
カスタムタグの定義

### ②<tag>タグ

<taglib>タグ内の<tag>タグはカスタムタグを定義するためのタグです。  
<tag>タグは一つ以上指定することができます。主な構成要素は以下のとおりです。

- <name>  
JSP 内で利用する際に指定するカスタムタグ名。
- <tag-class>  
カスタムタグに対応するタグハンドラクラス名。
- <body-content>  
empty — BODY は空でないといけない。  
JSP — デフォルト、JSP を解釈  
tag de pendent — BodyTagSupport で BodyContent から BODY を取得するとき  
に使う



---

# 索引

## A

application.....	27
autoFlush .....	25

## B

beanName .....	28
body-content.....	37
boolean isNew .....	11
buffer .....	25

## C

CATALINA__HOME .....	5
CGI .....	3
class.....	28
classes .....	5
config オブジェクト .....	27
contentType.....	25

## E

encodeURL.....	12
extends .....	24

## G

getAttribute()メソッド.....	10
getSession()メソッド.....	10
getter メソッド .....	32

## H

HTML.....	17
HTTP .....	8
HttpSessio.....	11
HttpSession インターフェース.....	10

## I

id .....	8
import .....	24
include ディレクティブタグ.....	26
info.....	25

## J

Java.....	14
JavaBeans .....	31
javax.servlet.jsp.JspPage.....	20
jsp:forward .....	29
jsp:getProperty.....	29
jsp:setProperty.....	29
jsp:useBean .....	23, 28
jsp:include .....	29
jsp:param .....	30
JSP.....	14, 15, 17, 18, 21
jspInit.....	20
jspService .....	20
jsp-version.....	37
JSP コンテナ.....	14, 20
JSP サーブレット .....	19, 20, 24
JSP ファイル .....	35
JSTL .....	34

## L

language.....	24
lib.....	5

## N

name .....	29, 37
------------	--------

O	
Object getAttribute .....	11
out オブジェクト .....	27

P	
page .....	28
pageContext オブジェクト .....	27
page ディレクティブタグ .....	23, 24
param .....	29
property .....	29

R	
request .....	28
request オブジェクト .....	26
response オブジェクト .....	26

S	
scope .....	28
session .....	25, 28
session オブジェクト .....	26
setAttribute() メソッド .....	10
setter メソッド .....	32
short-name .....	37
static 変数 .....	23
String .....	23

T	
tag .....	37
tag-class .....	37
taglib .....	37
taglib ディレクティブタグ .....	26
Time-Out .....	9
TLD ファイル .....	35, 37
tlib-version .....	37
Tomcat .....	14
Tomcat .....	2, 6
type .....	28
typeSpec .....	28

U	
URL Rewriting .....	12

V	
value .....	29
void invalidate .....	11
void removeAttribute .....	11
void setAttribute .....	11
voidsetMaxInactiveInterval .....	11

W	
WAR .....	4
web.xml .....	5
web.xml ファイル .....	35
webapps .....	5
WEB-INF .....	5
Webコンポーネント .....	4

X	
XML .....	21

あ	
アクションタグ .....	21, 28
アプレット .....	2

い	
インスタンス変数 .....	23

う	
Webアプリケーション .....	4

か	
隠れたコメント .....	22
カスタムタグ .....	15, 34, 36

き	
起動 .....	6

く	
クッキー .....	8, 9, 12
こ	
コメント .....	21, 22
さ	
サードパーティ .....	34
サブレット .....	2, 3, 10, 15
サブレットコンテナ .....	2
し	
式 .....	23
出力コメント .....	22
す	
スクリプトタグ .....	21, 23
スクリプトレット .....	23
ステートレス .....	8
スレッド・セーフ .....	20
せ	
セッション・オブジェクト .....	9, 10
セッション・トラッキング .....	8, 10

セッション ID .....	12
宣言 .....	23
た	
タグハンドラ・クラスファイル .....	35
タグライブラリ .....	34
て	
定義済みオブジェクト .....	21, 23, 26
停止 .....	6
ディレクティブ .....	23
ディレクティブタグ .....	21, 24
ふ	
プレゼンテーション・コンポーネント .....	4
フロントコンポーネント .....	4
ま	
マルチスレッド .....	20
め	
メソッド .....	23
る	
ルートディレクトリ .....	5