

JSP と開発環境の設定

変数と定数

第 1.0 版

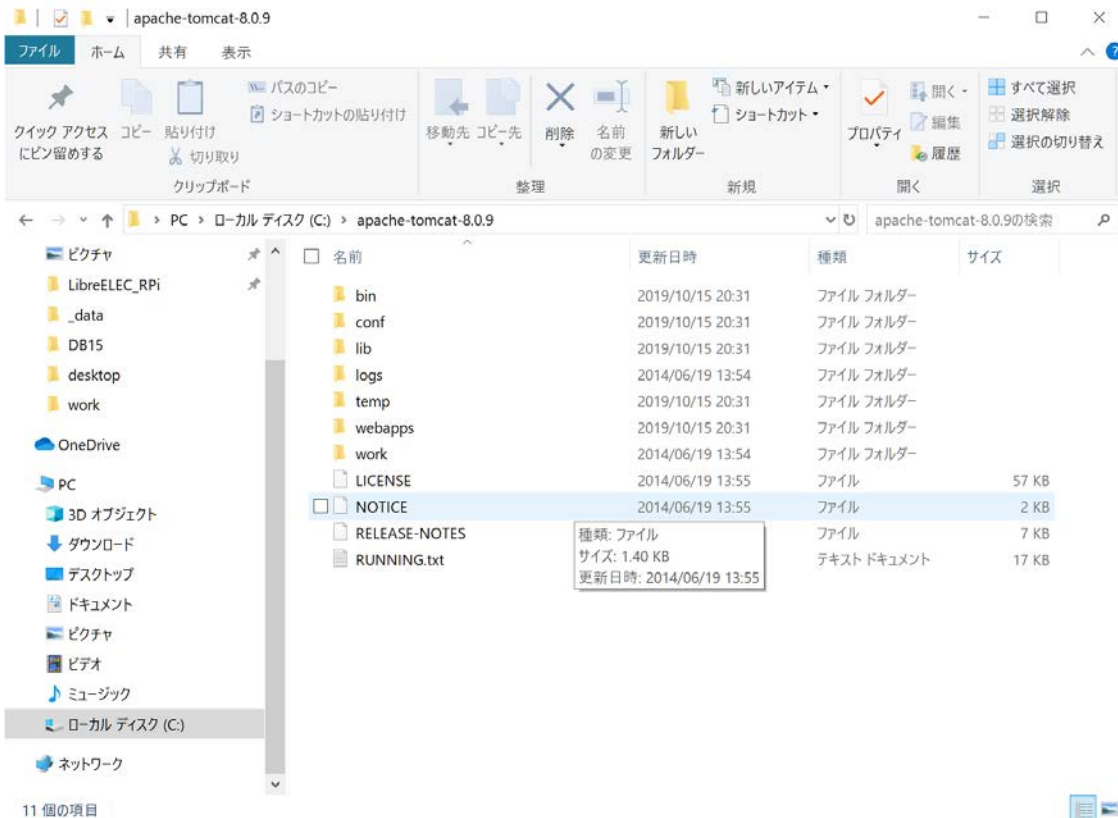
作 成 者	NH 中村広二
作 成 日	2019 年 10 月 11 日
最終更新日	2019 年 10 月 11 日

TOMCAT8 のインストール

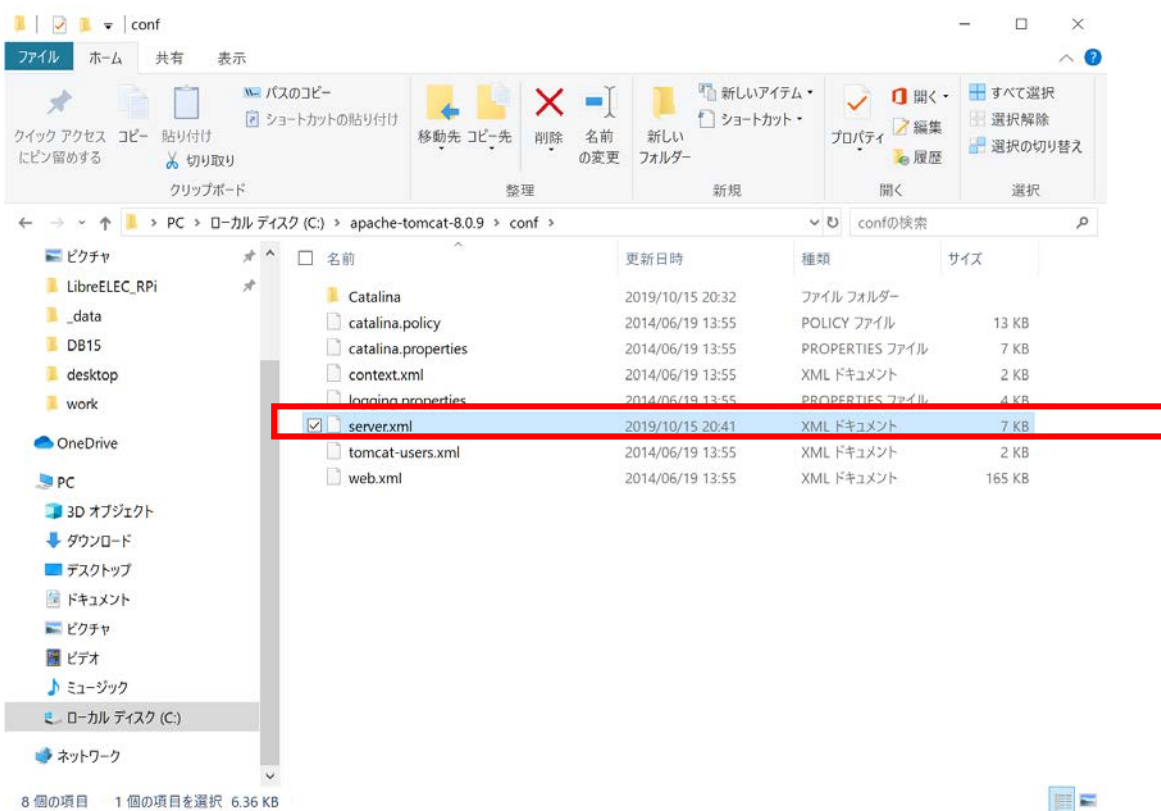
Apache Tomcat (アパッチ トムキャット) は、Java Servlet や JavaServer Pages (JSP) を実行するための Web コンテナ (サーブレットコンテナ、サーブレットエンジン) である。Apache License 2.0 を採用したオープンソースソフトウェア。

JSP (または Servlet) を動かすための必要なので次の手順で設定する。

(1) apache-tomcat-8.0.9.zip を、「c:\apache-tomcat-8.0.9」に解凍する。

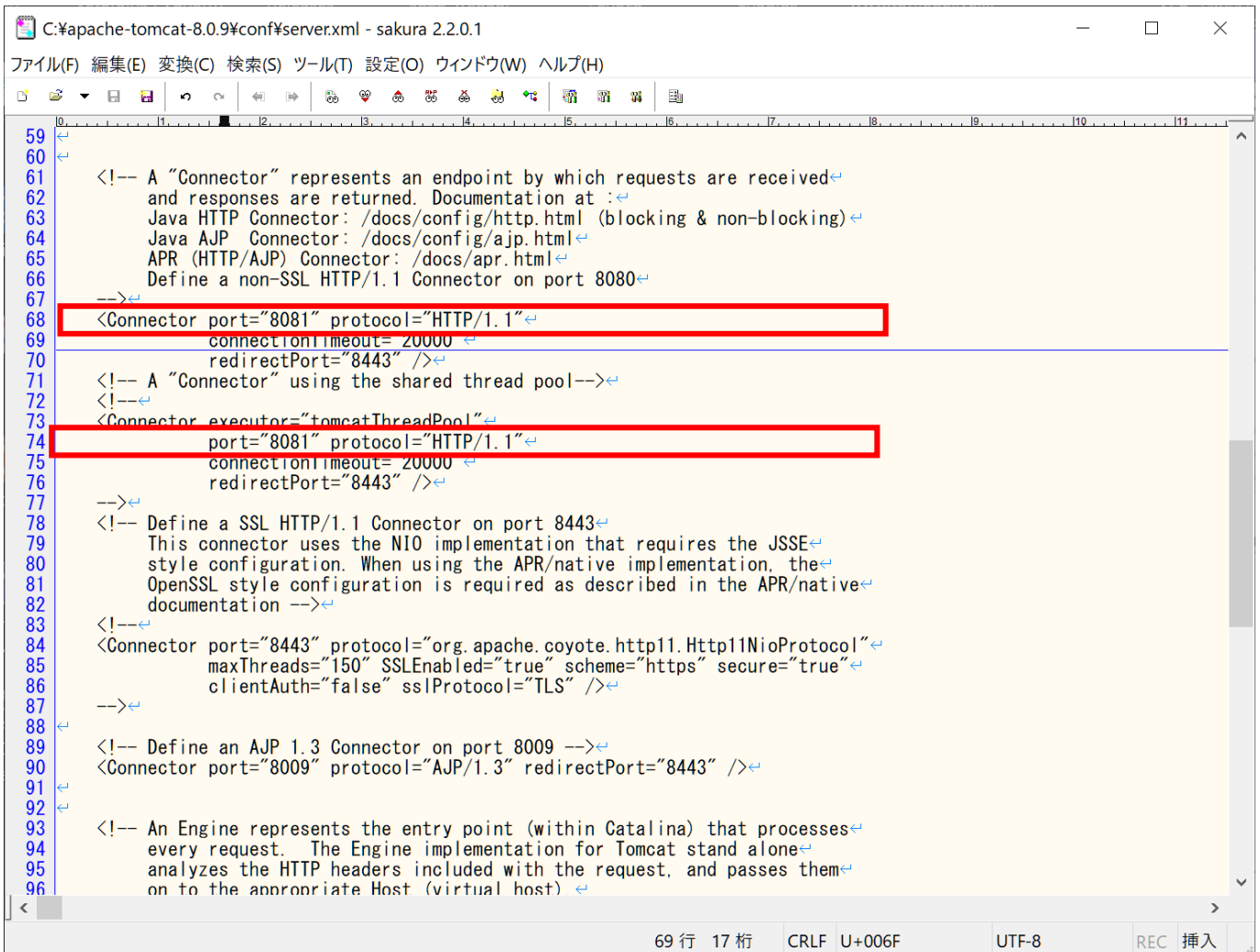


- (2) c:\¥apache-tomcat-8.0.9¥conf の server.xml をテキストエディター（さくらエディター等）で編集する。



(3) 開いた server.xml の 68 行目と 74 行目の 2 か所を、次のように編集し保存をする。

port=" 8080" → port=" 8081"

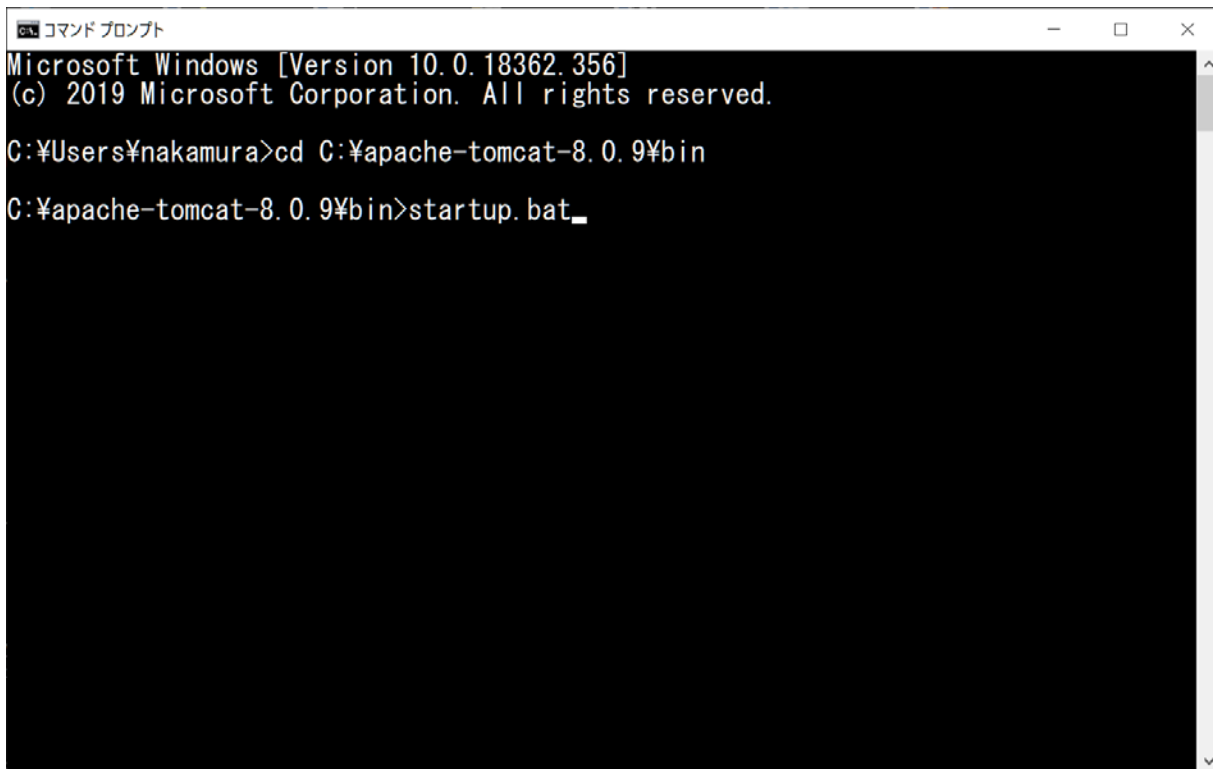


```
C:\apache-tomcat-8.0.9\conf\server.xml - sakura 2.2.0.1
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)
59
60
61 <!-- A "Connector" represents an endpoint by which requests are received
62 and responses are returned. Documentation at :
63 Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
64 Java AJP Connector: /docs/config/ajp.html
65 APR (HTTP/AJP) Connector: /docs/apr.html
66 Define a non-SSL HTTP/1.1 Connector on port 8080
67 -->
68 <Connector port="8081" protocol="HTTP/1.1"
69 connectionTimeout="20000"
70 redirectPort="8443" />
71
72 <!-- A "Connector" using the shared thread pool-->
73 <!--
74 <Connector executor="tomcatThreadPool"
75 port="8081" protocol="HTTP/1.1"
76 connectionTimeout="20000"
77 redirectPort="8443" />
78 -->
79 <!-- Define a SSL HTTP/1.1 Connector on port 8443
80 This connector uses the NIO implementation that requires the JSSE
81 style configuration. When using the APR/native implementation, the
82 OpenSSL style configuration is required as described in the APR/native
83 documentation -->
84 <!--
85 <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
86 maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
87 clientAuth="false" sslProtocol="TLS" />
88 -->
89 <!-- Define an AJP 1.3 Connector on port 8009 -->
90 <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
91
92 <!-- An Engine represents the entry point (within Catalina) that processes
93 every request. The Engine implementation for Tomcat stand alone
94 analyzes the HTTP headers included with the request, and passes them
95 on to the appropriate Host (virtual host)
96 -->
```

69 行 17 桁 CRLF U+006F UTF-8 REC 挿入

(4) コマンドプロンプトを開き、Tomcat を起動する。次のコマンドを入力する。

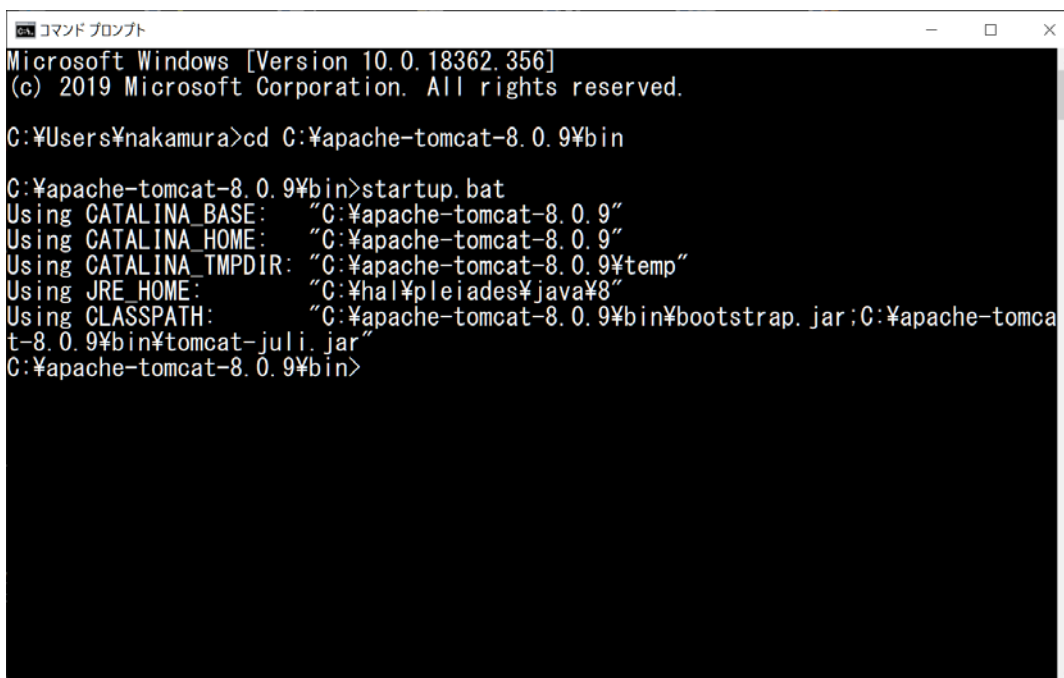
```
> cd C:\apache-tomcat-8.0.9\bin
> startup.bat
```



```
コマンド プロンプト
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\nakamura>cd C:\apache-tomcat-8.0.9\bin
C:\apache-tomcat-8.0.9\bin>startup.bat_
```

(5) Tomcat が正常に起動した場合、次ような画面が2画面表示する。
エラーが出ていないことを確認する。



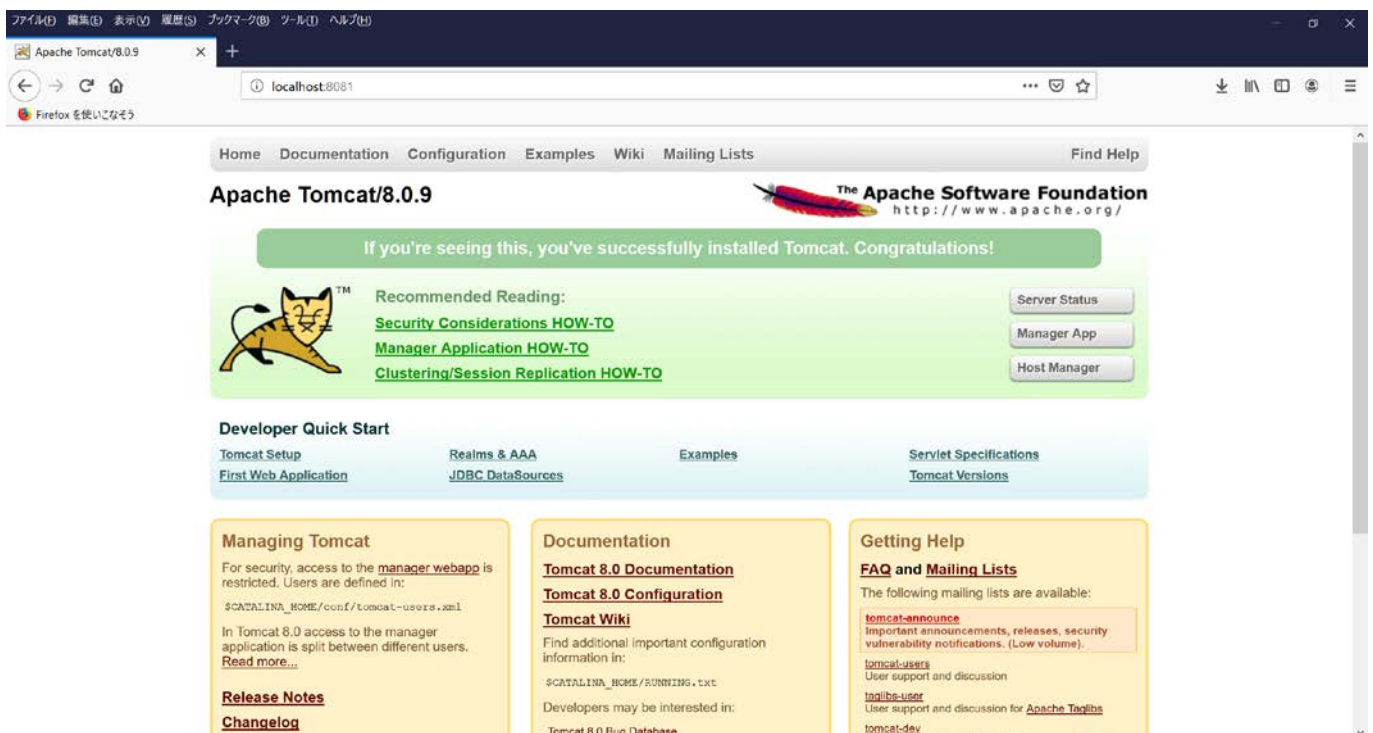
```
コマンド プロンプト
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\nakamura>cd C:\apache-tomcat-8.0.9\bin
C:\apache-tomcat-8.0.9\bin>startup.bat
Using CATALINA_BASE:   "C:\apache-tomcat-8.0.9"
Using CATALINA_HOME:   "C:\apache-tomcat-8.0.9"
Using CATALINA_TMPDIR: "C:\apache-tomcat-8.0.9\temp"
Using JRE_HOME:        "C:\hal\pleiades\java8"
Using CLASSPATH:       "C:\apache-tomcat-8.0.9\bin\bootstrap.jar;C:\apache-tomcat-8.0.9\bin\tomcat-juli.jar"
C:\apache-tomcat-8.0.9\bin>
```

```
Tomcat
15-Oct-2019 20:46:17.488 INFO [localhost-startStop-1] org.apache.catalina.startup.
p.HostConfig.deployDirectory Webアプリケーションディレクトリ C:\¥apache-tomcat-8.
0.9¥webapps¥host-manager を配備します
15-Oct-2019 20:46:17.550 INFO [localhost-startStop-1] org.apache.catalina.startup.
p.HostConfig.deployDirectory Deployment of web application directory C:\¥apache-t
omcat-8.0.9¥webapps¥host-manager has finished in 62 ms
15-Oct-2019 20:46:17.550 INFO [localhost-startStop-1] org.apache.catalina.startup.
p.HostConfig.deployDirectory Webアプリケーションディレクトリ C:\¥apache-tomcat-8.
0.9¥webapps¥manager を配備します
15-Oct-2019 20:46:17.613 INFO [localhost-startStop-1] org.apache.catalina.startup.
p.HostConfig.deployDirectory Deployment of web application directory C:\¥apache-t
omcat-8.0.9¥webapps¥manager has finished in 63 ms
15-Oct-2019 20:46:17.613 INFO [localhost-startStop-1] org.apache.catalina.startup.
p.HostConfig.deployDirectory Webアプリケーションディレクトリ C:\¥apache-tomcat-8.
0.9¥webapps¥ROOT を配備します
15-Oct-2019 20:46:17.660 INFO [localhost-startStop-1] org.apache.catalina.startup.
p.HostConfig.deployDirectory Deployment of web application directory C:\¥apache-t
omcat-8.0.9¥webapps¥ROOT has finished in 47 ms
15-Oct-2019 20:46:17.660 INFO [main] org.apache.coyote.AbstractProtocol.start St
arting ProtocolHandler ["http-nio-8081"]
15-Oct-2019 20:46:17.675 INFO [main] org.apache.coyote.AbstractProtocol.start St
arting ProtocolHandler ["ajp-nio-8009"]
15-Oct-2019 20:46:17.691 INFO [main] org.apache.catalina.startup.Catalina.start
Server startup in 1376 ms
```

(6) ブラウザの URL に次のアドレスを入力して、次の画面が開いたら Tomcat インストール成功。

http:// http://localhost:8081/



【メモ】

JAVA_HOME を設定してない場合、以下のエラーがでます。

```
Neither the JAVA_HOME nor the JRE_HOME environment variable is defined  
At least one of these environment variable is needed to run this program
```

この場合は、コマンドプロンプトから次のコマンドを入力し startup.bat をキックする。

【例】

```
>SET JAVA_HOME=C:\Program Files\Java\jdk1.8.0_60
```

Hello Jsp!

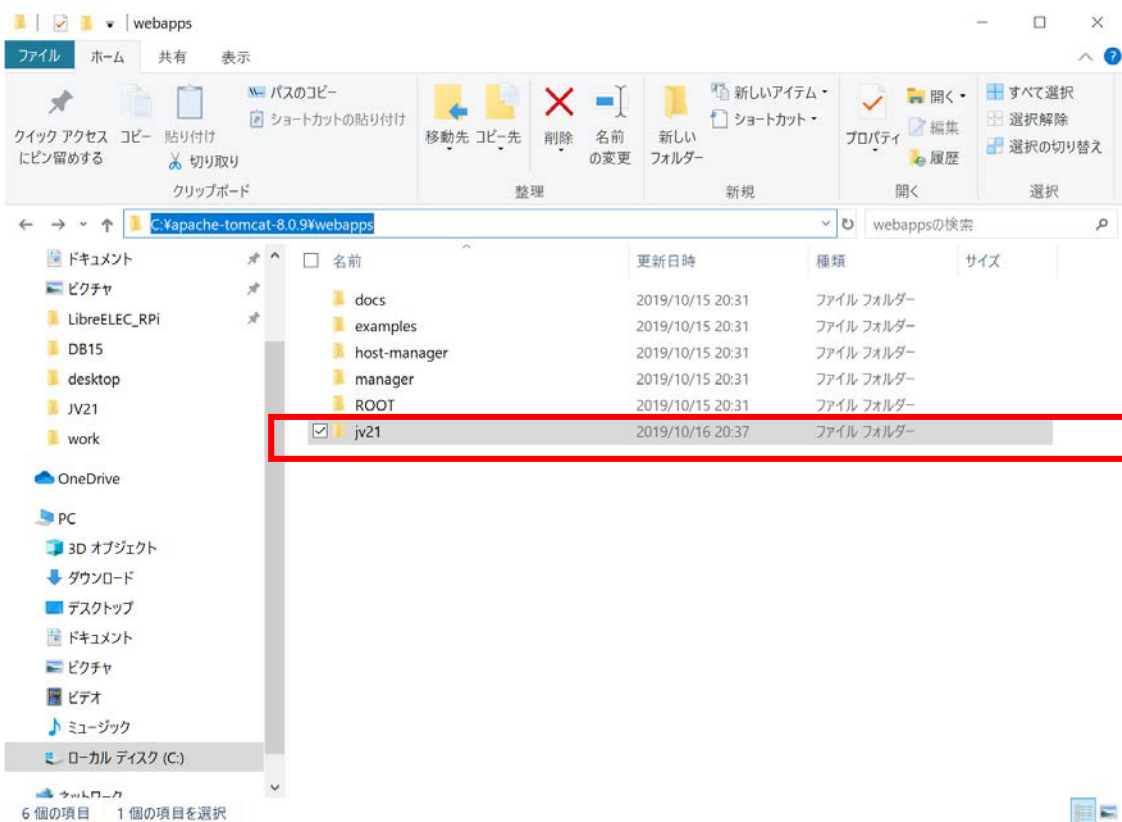
(1) 次の Jsp をエディタで作成する。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<html>
  <head>
    <title>Hello World!</title>
  </head>
  <body>
    <h1>Hello Jsp!</h1>

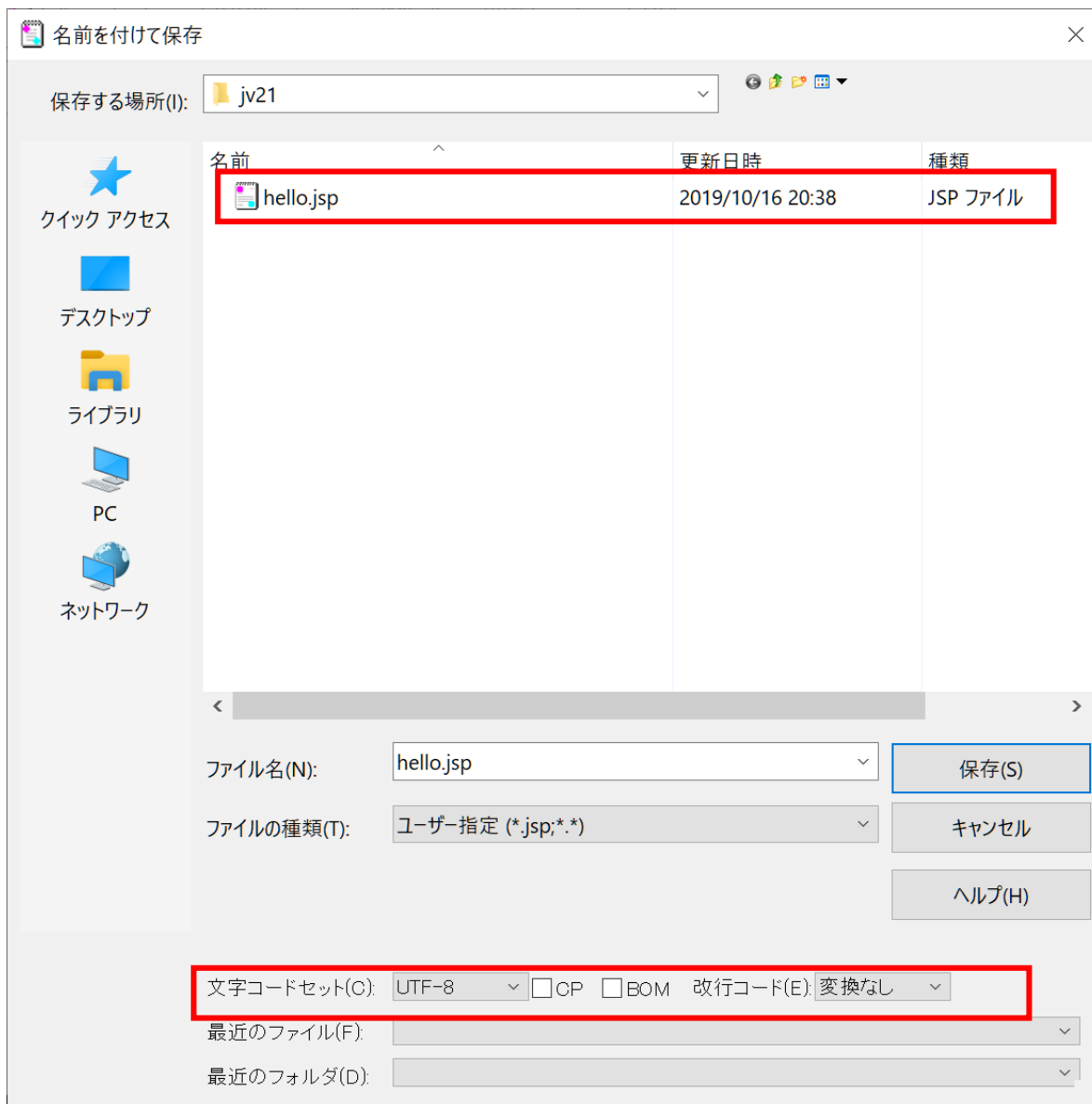
    <%
      out.println(new java.util.Date());
    %>

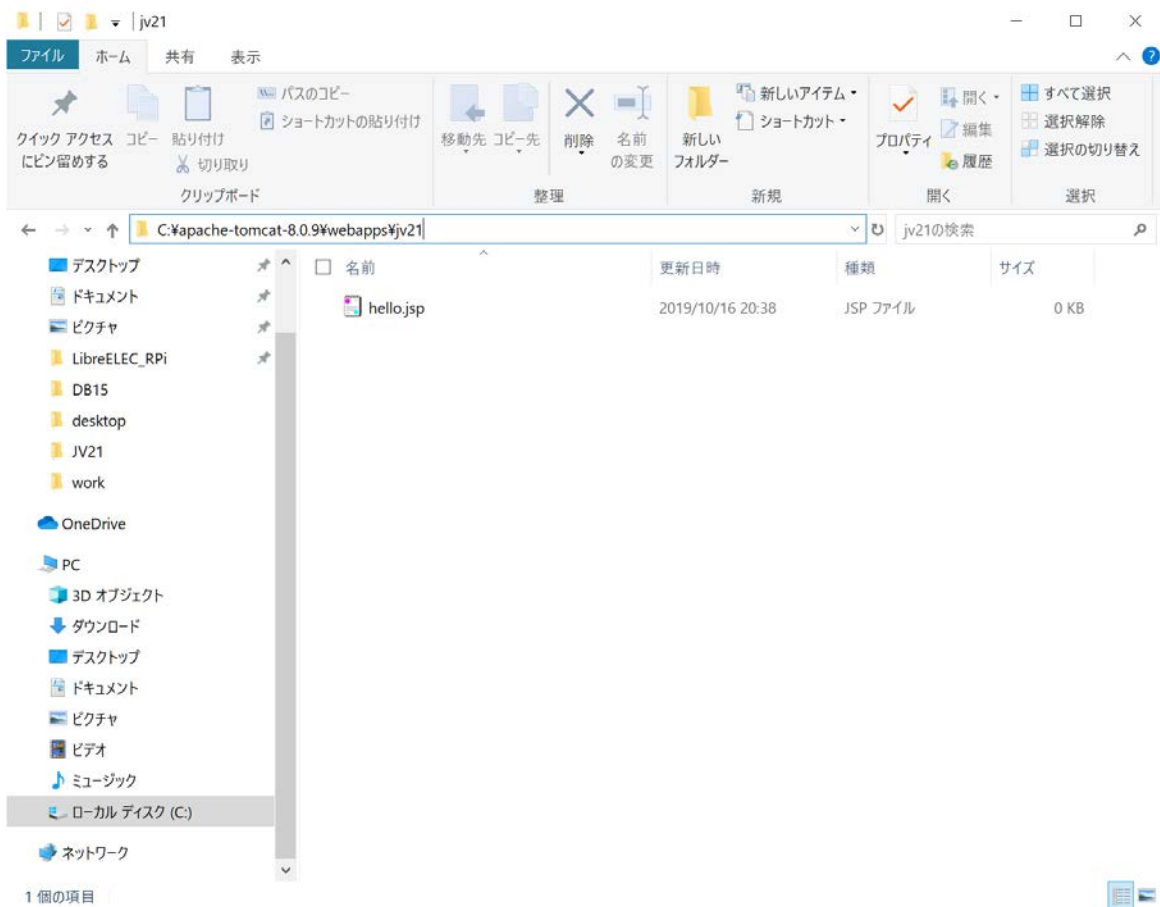
  </body>
</html>
```

(2) c:\¥apache-tomcat-8.0.9¥webapps 配下に「jv21」という名前でフォルダを作成する。



(3) 「hello.jsp」と名前を付けて保存する。文字コードを「UTF-8」にする。





(4) Tomcat をコマンドプロンプトから起動する。

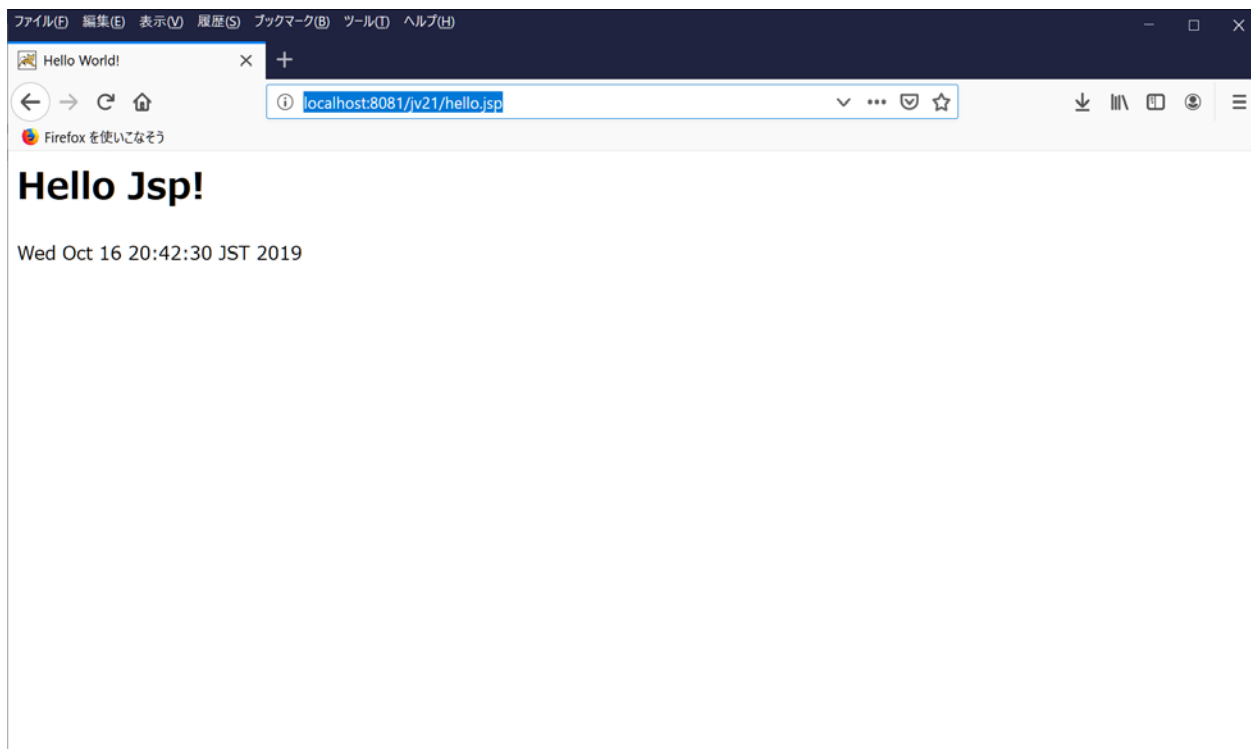
```
コマンド プロンプト
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\nakamura>cd C:\apache-tomcat-8.0.9\bin

C:\apache-tomcat-8.0.9\bin>startup.bat
Using CATALINA_BASE: "C:\apache-tomcat-8.0.9"
Using CATALINA_HOME: "C:\apache-tomcat-8.0.9"
Using CATALINA_TMPDIR: "C:\apache-tomcat-8.0.9\temp"
Using JRE_HOME: "C:\hal\pleiades\java\8"
Using CLASSPATH: "C:\apache-tomcat-8.0.9\bin\bootstrap.jar;C:\apache-tomcat-8.0.9\bin\tomcat-juli.jar"
C:\apache-tomcat-8.0.9\bin>
```

- (5) ブラウザを起動して次の URL を入力しエンターを押す。
「Hello JSP!」と表示したら成功。

`http://localhost:8081/jv21/hello.jsp`



基本データ型（プリミティブデータタイプ）

Java ではデータを格納する場所を「変数」と呼ぶ。Java では長さが決まった以下の 8 種類の変数を使用する。これを基本データ型（プリミティブデータタイプ）と言う。

名称	ビット長	内容	値の範囲	デフォルト値
boolean	1	真偽値	true または false	false
char	16	文字 (Unicode)	¥u0000~¥uFFFF (*)	¥u0000
byte	8	符号付き整数	-128~127	0
short	16	符号付き整数	-32768~32767	0
int	32	符号付き整数	- (2 の 31 乗) ~ (2 の 31 乗) -1	0
long	64	符号付き整数	- (2 の 63 乗) ~ (2 の 63 乗) -1	0
float	32	浮動小数点数	単精度浮動小数点数	0.0
double	64	浮動小数点数	倍精度浮動小数点数	0.0

(*) ¥uXXXX は 16 進表示

上記 8 種の内容と値の範囲は記憶すること。整数型はいずれも正の数の値の範囲が負の数より 1 だけ少なくなっている。理由は 0 が正の数側に入るから。

ラッパークラス

Java では、前述の基本データ型が用意されている。数値や文字を格納したり演算子と組み合わせる場合であれば基本データ型を使えば十分。ただ基本データ型にはクラスのようにメソッドは用意されていないので、基本データ型に対して操作を行ったりすることはできない。

そこで基本データ型の値を持つことができ、様々な便利なメソッドを用意したクラスが用意されている。それがラッパークラスと呼ぶ。基本データ型から対応するラッパークラスのオブジェクトを作成し、そのオブジェクトに対して用意されたメソッドを利用することで例えば数値から文字列への変換といった処理が可能になる。

基本データ型	ラッパークラス
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

int 型のラッパークラスは Integer クラス、float 型のラッパークラスは Float クラス。基本的には基本データ型の先頭文字を大文字にしたものがラッパークラスの名前だが、char 型と int 型だけは異なるので注意すること。

型宣言

変数を使うには、まず変数の型宣言を行う。型宣言は次のようにデータ型名の後に変数名を書くことによって行う。変数の型宣言は変数を使う前ならばプログラム中のどこにあってもかまわない。

```
int a;           // int (32 ビット整数) 型の a という名前の変数を宣言
```

Java では文の最後にセミコロン (;) を付ける。
セミコロンがあれば 1 行に複数の文を書いてもかまわない。
またスラッシュ (/) を 2 つ連続して書くと、それ以降がコメント文になる。

代入

変数に値を代入するときは次のように等号 (=) を使う。

```
a = 10;  
b = 12.3;  
c = b + 4.5;
```

初期化は、型宣言と同時に値を設定することができる。

```
int a = 10;  
double b = 12.3;
```

Java では値が設定されていない変数を参照しようとするエラーになる。このエラーはコンパイルの段階で指摘してくれる。ただし、変数の宣言場所によっては、デフォルト値が設定されることがあり、この場合にはエラーにはならない。

型変換

変数の演算は同じデータ型どうしで行うのが望ましい。実用上はそれでは済まないときがある。そのために以下のような型変換機能が用意されている。

- (a) 演算時の変換（演算時に互いの型を合わせる）
 - 1. 表現の範囲が大きな方の型に合わせる。 大小関係は以下の通り。
double > float > long > int
 - 2. 32 ビットの float の方が 64 ビットの long より大きいことに注意。
int より小さい byte、char、short は int に変換される。
- (b) 代入時の変換（右辺の型を左辺の型に変換する）
 - 3. 左辺の型の表現範囲が右辺の型より大きい場合はそのまま代入される（拡張変換）。
int → float などの代入ではエラーにはならないが、有効数字は落ちることがある。
 - 4. 逆の場合はコンパイルエラーとなる。ただし型変換を明示的に指定すれば、はみ出した部分が切り捨てられて代入される（縮小変換）。
short と char 間および byte と char 間はいずれも縮小変換。（符号情報の欠落などがあるため）

上記 4. で述べた、明示的な型変換をキャストという。キャストは変数名の前に、変換先の型名を括弧でくくることで行う。キャストの例を示す。

```
int id = 65539;
short sd;
double dd = 3.14;
× sd = id;           // 縮小変換 (32→16 ビット) なのでコンパイルエラーとなる
sd = (short)id       // キャストしたので OK。上位が切り捨てられ、sd は 3 になる
× boolean b = (boolean)id
// 何でもキャストできるわけではない。例えば boolean 型へはキャストできない
id = (int)dd;        // 小数点以下が切り捨てられ、id は 3 となる
```

文の前に × がついているのはエラーになることを示す。

※ Java は型変換に大変「うるさい」言語。厳しくチェックすることで、うっかりミスによるバグを防止しようとの意図がある。C → C++ → Java と新しい言語が作られるたびに、規則は厳格化してきた。

char 型の値

char は文字ですが、算術演算を行うときは符号無し 16 ビット型として扱われる。

```
char cd = (char)65534;
int id = cd;    // id は 65534 になる
short sd = (short)cd; // 先頭ビットを符号とみなし sd は -2 になる
```

boolean 型の値

boolean 型の値は true か false のいずれか。boolean 型は他のどのデータ型へも型変換できない。逆に他のどの基本データ型からも型変換できない。

文字列型

文字列を格納する変数は String。String を使うと文字列を簡単に操作することができる。
文字列を + で結ぶと文字列を結合できる。文字列と数値を + で結ぶと数値を文字に変換してから結合する。
以下の例。

```
String ss = "ABC";      // String（文字列）型の変数 ss を宣言し値 ABC を代入
ss = ss + "DEF";        // 文字列結合がおこり ss は"ABCDEF"になる
× ss = 111;             // 数値を文字列に代入するのでコンパイルエラー
ss = "" + 111;          // 文字列に変換してから代入するので ss は"111"
ss = 111 + 222 + "";    // 整数値の加算後、文字列結合するので ss は"333"
ss = "" + 111 + 222;    // 文字列結合がおこるので ss は"111222"
ss.length();           // 文字列の長さを int 型で返す
```

Java では文字は Unicode で記述される。Unicode は JIS 漢字コードやそれを変形した ShiftJIS、EUC などとは全く異なった文字コード体系ですので規則的な変換はできない。

定数

定数は値の決まった数や文字のことでリテラルとも呼ぶ。
定数には整数、浮動小数点数、文字、文字列などがある。整数値は、10 進数だけでなく、8 進数や 16 進数で表現することもできる。

```
0123;    // 0 で始めると 8 進数。10 進表示では 64+2*8+3 で 83 になる
0x1234;   // 0x で始めると 16 進数。10 進表示では 4660 になる
```

定数の型（整数）

定数にも変数と同じように型がある。単に整数値を書くと int 型とみなされる。int 型ではなく long 型であることを明示したいときには、数値の後ろに L または l（小文字の L）を付ける。以下の 1 行目は右辺の数字が int 型と見なされ、代入前の時点でエラーになる。2 行目のように数字の後ろに L を付ければエラーとならない。

```
× long ld = 12345678900; // 数値が int の範囲を越えるのでコンパイルエラー
long ld = 12345678900L;  // long 型であることを明示したので OK
```

浮動小数点数

小数点付きの数値はデフォルトで double とみなされる。double 型ではなく float 型であることを明示したいときには、数値の後ろに F または f を付ける。浮動小数点数は指数形式であらわすこともできる。

```
× float fd = 12.34;      // 数値は double 型なのでコンパイルエラー
float fd = 12.34f;       // float 型であることを明示すれば OK
double dd = 1.2e-3;      // 指数形式で指定。dd は 0.0012 になる
```


文字定数と文字列定数

文字をシングルクォート（'）で囲むと文字定数になる。一方ダブルクォート（"）で囲むと文字列定数になる。前者は1文字だけ、後者は複数文字を扱うので違いに注意する。

```
char cd1 = 'A' // 半角文字の Unicode は JIS7 ビットコードと同じ
char cd2 = '漢' // 漢字 1 字も文字定数
× char cd3 = 'AB' // これは文字列なのでコンパイルエラー
× char cd4 = "A" // 文字列を文字に代入するのでコンパイルエラー
```

エスケープ文字（¥）

改行やタブコードなどは次のように ¥ と特定の文字を組み合わせで表現する。

```
String s1 = "ABC¥n"; // ABC（改行）を示す。n は new line の略
String s2 = "A¥tB"; // A（タブコード）B を示す
char c1 = '¥'; // シングルクォート（'）を示す
char c2 = '¥"'; // ダブルクォート（"）を示す
char c3 = '¥¥'; // ¥自身を示す
```

識別子の命名規則

変数の名前の付け方には次のような規則がある。文字の長さの制限はない。これらの規則は変数名だけでなく、後述のメソッド名やクラス名などの命名規則としても適用される。これらを他と識別する名前であるとして識別子と呼ぶ。

- ① 使える文字は英字（大文字、小文字）、数字、\$、下線（_）。数字は先頭には使えない
- ② 予約語（キーワード）は使えない（予約語リストを後述）
- ③ 英字の大文字と小文字は区別される（Data と data は全く別の変数になる）
- ④ 漢字や全角かな等も Unicode なので使える（実際にはあまり使われない）

※ Java ではハイフンが使えない。かわりに下線が使える。

慣用規則

以上の他に慣用として次の規則も広く使われる。これらは言語仕様ではないので、従わなくてもエラーにはならない。

- ⑤ 基本的に英小文字を使用する
- ⑥ 変数名とメソッド名は英小文字で始める
- ⑦ クラス名は英大文字で始める
- ⑧ 2 単語目の先頭は英大文字とする（例：myData（変数名）、MyClass（クラス名））
- ⑨ 定数値を入れるときは全部を大文字にすることが多い（例：MAXVALUE）

予約語リスト

Java の予約語を示す。

基本データ型とその値 (9 語)

byte, char, short, int, long, float, double, boolean, void

クラス関連 (10 語)

class, interface, extends, implements, this, super, new, instanceof, import, package

修飾子 (11 語)

public, protected, private, final, static, abstract, native*,
synchronized, volatile*, transient*, strictfp*

制御構造関連 (11 語)

if, else, for, while, do, switch, case, default, break, continue, return

例外処理関連 (5 語)

try, catch, finally, throw, throws

その他 (2 語)

assert, enum

使用されないが予約されている (2 語)

const*, goto*

予約語ではないがリテラルとして定義されていて識別子として使えない (3 語)

true, false, null

Jsp を用いた Java の演算例

- (1) 次の Jsp をエディタで作成し「count.jsp」と名前を付けて保存する。

```
<%@ page contentType="text/html; charset=Shift_JIS" %>

<html><head><title>カウント</title></head>
<body>
<h1>カウント</h1>

<%!
int count = 0;
%>

<%
int localcount = 0;

count++;
localcount++;
%>

<p>
スクリプトレット内のカウントは<%= localcount %>です。
</p>

<p>
変数宣言で作成したカウントは<%= count %>です。
</p>

</body>
</html>
```

- (2) Tomacat 上の jv21 フォルダ内に配置（デプロイ）して次の URL で動作を確認する。

<http://localhost:8081/jv21/count.jsp>

<END>