# Java を使って Web アプリをつくる

BlackNihonkai

更新日：2024 年 9 月 17 日 1 時 22 分

## 1　開発環境のセットアップ

まずは、開発環境を構築する。JDK のインストールも必要となる。

### 1.1　Pleiades のダウンロード

Java を使ってアプリケーションを作成する場合には、Eclipse を用いるのが一般的である。Web アプリケーションを作る場合には、Pleiades を用いて開発するのが便利らしい。Pleiades を用いる場合には Eclipse も一緒にインストールされ、Eclipse での開発も便利になる。

Pleiades のインストールするには、`https://willbrains.jp/` にアクセスし、図 1 の”Pleiades ALL in One”をダウンロードする。(原則として、) 最新版のもの (図 1 の場合は「Eclipse 2024」) をクリックして、図 2 の画面に遷移する。

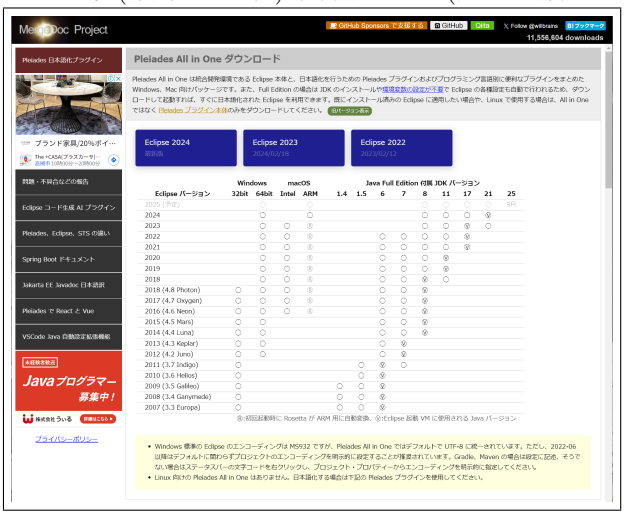


図 1　Pleiades All in One のバージョン選択


図 2　Pleiades All in One のダウンロード

### 1.2　Pleiades のインストール

前項の手順でダウンロードした”Pleiades All in One”は、zip の状態であるため zip を回答するだけでインストールが始まる。

## 2　Web アプリケーションをつくる

ここからは、別のリポジトリで挑戦中のナンプレアプリをつくる。自分へのメモ代わりである。

### 2.1　サーブレットと JSP

Java を使って Web アプリを作る場合には、サーブレット (Servlet) と JSP なるものを使うとよいらしい。

Servlet と JSP は基本的には同じものである。しかし、プログラマーとデザイナーが仕事の分担をできるように、同じようなものが二つ存在している。一般的に、プログラマーは (Java などの) プログラミングは得意でも Web ページのデザインは苦手なことがある。デザイナーは (HTML などによる)Web ページのデザインは得意でもプログラミングは苦手なことが

ある。そこで、プログラマーが Servlet でプログラミングに専念し、デザイナーが JSP で Web ページの作成に専念するという形をとる。

Java で Web アプリを作る場合には、一つの Web アプリ (正確には、一つの Web ページ) に対して、Servelet と JSP をペアで用意する。Servlet がユーザの要求 (リクエスト) を受け取って処理を行い、その応答 (レスポンス) となる Web ページを JSP で返す。

例)

ユーザ ID とパスワードを入力してログインを行う Web アプリを作るとする。この Web アプリを、LoginServlet(という名前の Servlet) と login.jsp(という名前の JSP) のペアから構成することにする。するとまず、Web ブラウザの URL 欄に LoginServlet を指定すると、LoginServlet に対して GET リクエストが送られる。次に Servlet は、このリクエストに対する処理として、login.jsp へフォワードを行う。ただし、フォワードとは、同じ Web アプリ内で処理を先に進めることをいう。

次に login.jsp は、レスポンスとして、ログインページの HTML を返す。このログインページにユーザ ID・パスワードを入力して”ログイン”ボタンをクリックすることで、LoginServlet に Post リクエストが送られる。最後に LoginServlet は、ユーザ ID とパスワードの検証を行い、ログイン成功か失敗かの Web ページにリダイレクトする。

これが Servlet と JSP の役割分担の例となる。

## 2.2 実装 (ひとまずの実装)

### 2.2.1 Eclipse でプロジェクトを作成

まず、Eclipse を起動して、”File > New > Dynamic Web Project”を選択する。日本語環境においては、”Dynamic Web Project”の部分が”動的 Web プロジェクト”などの表記になっているだろう。

動的 Web プロジェクトのウィンドウで、プロジェクト名を入力 (例：SudokuWebApp)、Dynamic web module version を選択する ('24/9/15 時点では”3.1”推奨)。Finish をクリックしてプロジェクトを作成する。

### 2.2.2 サーブレットの作成

src フォルダ内の”Java Resources > src”を右クリックし、”New > Servlet”を選択する。開いたウィンドウで、サーブレット名を入力する (例：SudokuServlet)。DoGet メソッドを利用するようにチェックを入れて、Finish をクリックする (ただし、執筆時点ではこの設定項目がなかったような気がする)。

今回はナンプレの盤面を仮に設定し、これを表示するまでのサーブレットを作成しコード 1 に示す。※ただし、このコードは ChatGPT が吐き出したものであるが、この項を執筆している時点では動作した。

コード 1 ”SudokuServlet”のコード例

```
1   import java.io.IOException;
2   import javax.servlet.ServletException;
3   import javax.servlet.annotation.WebServlet;
4   import javax.servlet.http.HttpServlet;
5   import javax.servlet.http.HttpServletRequest;
6   import javax.servlet.http.HttpServletResponse;
7
8   @WebServlet("/SudokuServlet")
9   public class SudokuServlet extends HttpServlet {
10      private static final long serialVersionUID = 1L;
11
12      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
13          // 数独ボードを作成する簡単な例
14          int[][] board = {
15              {5, 3, 0, 0, 7, 0, 0, 0, 0},
16              {6, 0, 0, 1, 9, 5, 0, 0, 0},
```

```
                {0, 9, 8, 0, 0, 0, 0, 6, 0},
                {8, 0, 0, 0, 6, 0, 0, 0, 3},
                {4, 0, 0, 8, 0, 3, 0, 0, 1},
                {7, 0, 0, 0, 2, 0, 0, 0, 6},
                {0, 6, 0, 0, 0, 0, 2, 8, 0},
                {0, 0, 0, 4, 1, 9, 0, 0, 5},
                {0, 0, 0, 0, 8, 0, 0, 7, 9}
            };

            request.setAttribute("sudokuBoard", board);
            request.getRequestDispatcher("sudoku.jsp").forward(request, response);
        }
    }
```

### 2.2.3 JSP ファイルの作成

WebContent フォルダ内で”New > JSP File”を選択し、ファイル名を入力する (例：sudoku.jsp)。コード 2 のように、サーブレットから送られてきたナンプレボードを表示する JSP ページを作成する。※このコードについても、ChatGPT が吐き出したものであり、この項を執筆している時点では動作した。また、”board”が”null”出ないかの確認をするように実装されている。

コード 2 ”sudoku.jsp”のコード例

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Sudoku</title>
</head>
<body>
    <h1>Sudoku Board</h1>
    <table border="1">
        <%
            int[][] board = (int[][]) request.getAttribute("sudokuBoard");
            if (board != null) {
                for (int i = 0; i < 9; i++) {
                    out.println("<tr>");
                    for (int j = 0; j < 9; j++) {
                        out.println("<td>" + (board[i][j] == 0 ? "" : board[i][j]) + "</td
                            >");
                    }
                    out.println("</tr>");
                }
            } else {
                out.println("<p>Error: Sudoku board not found!</p>");
            }
        %>
    </table>
</body>
</html>
```

### 2.2.4 Web アプリケーションの実行

プロジェクトを右クリックし、”Run As > Run on Server”を選択する。ただし、初回の場合はサーバの設定などが必要となる。SudokuServlet が”http://localhost:8080/SudokuWebApp/SudokuServlet”で実行され、ナンバープレースのボードが表示される。実行時に自動的にブラウザが起動された場合には、URL を正しく入力されていない場合があるので、その

ときは自分で URL 欄を書き直す必要がある。

## 2.3 改善

### 2.3.1 見た目の改善１

まずは、JSP ファイルをもとに表示されるナンプレの盤面を、より見慣れた感じにする。ナンプレの盤面を実現するために CSS を設定する。CSS を設定してナンプレ盤面を再現するコードをコード 3 に示す。

※相変わらず、ChatGPT が吐き出したものであり、この項を執筆している時点では動作することを確認した。

コード 3　ナンプレの盤面を再現

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Sudoku</title>
    <style>
        table {
            border-collapse: collapse;
            margin: 20px auto;
            font-size: 20px;
            text-align: center;
        }
        td {
            width: 40px;
            height: 40px;
            border: 1px solid black;
            vertical-align: middle;
        }
        /* ３×３のブロックを区切る太い線*/
        td.block-border-top {
            border-top: 3px solid black;
        }
        td.block-border-left {
            border-left: 3px solid black;
        }
        td.block-border-right {
            border-right: 3px solid black;
        }
        td.block-border-bottom {
            border-bottom: 3px solid black;
        }
    </style>
</head>
<body>
    <h1>Sudoku Board</h1>
    <table>
        <%
            int[][] board = (int[][]) request.getAttribute("sudokuBoard");
            if (board != null) {
                for (int i = 0; i < 9; i++) {
                    out.println("<tr>");
                    for (int j = 0; j < 9; j++) {
                        // 各セルにｃｓｓクラスを割り当てることで３×３の区切り線を作成
                        String cssClass = "";
                        if (i % 3 == 0) cssClass += "␣block-border-top";
                        if (i == 8) cssClass += "␣block-border-bottom";
```

```
46                              if (j % 3 == 0) cssClass += "␣block-border-left";
47                              if (j == 8) cssClass += "␣block-border-right";
48
49                              out.println("<td␣class='" + cssClass.trim() + "'>" + (board[i][j]
                                    == 0 ? "" : board[i][j]) + "</td>");
50                          }
51                          out.println("</tr>");
52                      }
53                  } else {
54                      out.println("<p>Error:␣Sudoku␣board␣not␣found!</p>");
55                  }
56              %>
57          </table>
58      </body>
59      </html>
```