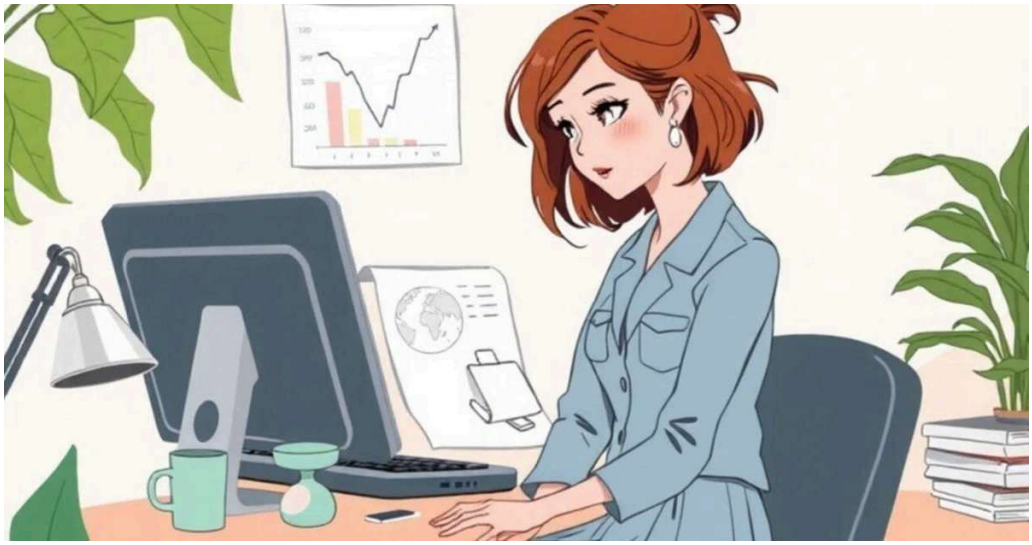

MVCモデルによるWebアプリケーション開発（No11.２進10進変換器）-EE8

office・M
2024年11月23日 18:13

¥1,000 •••

JavaEE8（JSP・Servlet）の環境でWebアプリケーションの開発方法を学ぶ講座をシリーズで提供しています。今回は入力と出力で画面が遷移しない題材（2進10進変換器）をMVCモデルで実装してみます。

2024年9月よりECLIPSEのバージョンを最新版（Version: 2024-06 (4.32.0)）に変更しました。

目次

開発概要

JavaEE8アプリケーション・サーバ（Tomcat）を利用して開発を行うということを前提に実装を進めていきます。開発ツールには統合開発環境のEclipseを用いることにします。

題材として2進数と10進数を相互に変換するWebアプリケーションを作成します。入力画面のラジオボタンで変換する方向（2進数から10進数かその逆か）を選択し変換したい数値を入力すると、同じ画面内に変換結果を表示します。入力画面と変換結果の出力画面には同じJSPを使うため、GETリクエスト時にもBeanの実体化が必要になります。

JavaEE8環境ですので使用言語は当然Java（JSP・Servlet）となり、設計技法にはMVCモデルを使います。MVCモデルとはWebアプリケーションの構成をModel（業務ロジック）ーView（表示）ーController（制御）に分割して設計する技法です。３つのモデルに役割分担することで部品化が促され、ひいてはチーム開発に貢献します。

実装していく過程でEclipseの利用方法が不明な場合は、別途メール等でお知らせください。この講座では、作成済みの仕様と設計書から「**JavaEE8環境でのMVCモデルの実装の仕組みを学ぶ**」ことを目的とします。

開発方針

Webアプリケーションの開発環境には図１のようにEclipse2024を用います。

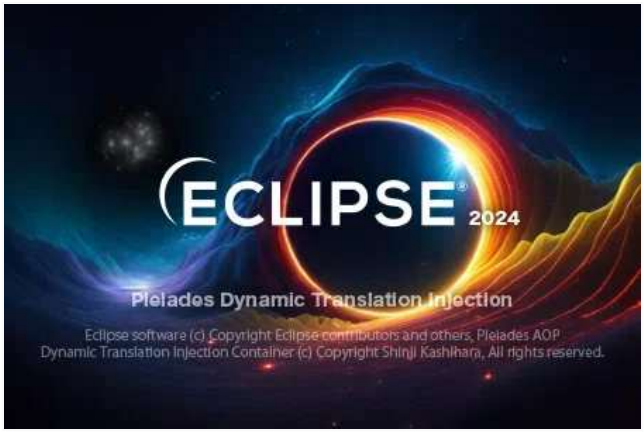

図1. 開発環境 Eclipse 2024

Spring等のフレームワークは使用しません。これは、MVCモデルにおけるhttpプロトコルの処理の実装を直に学んでいただきたいためです。

開発に必要な仕様と最小限の設計ドキュメントは提示します。この情報をもとに、まずは、自分で試行錯誤しながら実装してみてください。

設計仕様

【２進10進変換器の仕様】
以下の条件を具現化するWebアプリケーションの作成を行います。
（条件）
・ラジオボタンで2進数から10進数への変換かその逆かを選択します。
・数値フィールドに変換したい数値を入力します。
・「変換ボタン」をクリックすると同じ画面内に変換結果を表示します。
・変換可能な範囲を外れた数値を入力した場合、数字以外の文字を入力した
　場合はエラーメッセージを表示します。
・入力する10進数は整数値です。
（変換範囲）
2進数10進数の相互変換可能な数値はint型の取りうる範囲とします。Javaのint型は**32ビット**になるので**10進数では-2147483648～2147483647**の値を変換することが可能になります。また、Javaでは負数を表現するために先頭ビットが符号ビットとなるため**2進数は31ビット以内で入力**することにします（2進数で負の数の変換は考慮しません）。

この範囲を超えた入力でエラーメッセージを出力するようにします。

【Model（処理ロジック）の設計方針】
以下のModelを実装します。
BinaryDecimalBeanクラス：
・Beanの仕様に従い実装する
・２進数を10進数に変換するメソッドを実装する
・10進数を２進数に変換するメソッドを実装する
BinaryValidatorクラス：

・通常クラスで実装する
・入力値が２進数として適切であるか評価するメソッドを実装する
DecimalValidatorクラス：
・通常クラスで実装する
・入力値が10進数として適切であるか評価するメソッドを実装する

外部設計

接続urlは**http://localhost:8080/converter/BinaryDecimal**とします。よってEclipseの動的Webプロジェクトの名前は**converter**となり、サーブレットのurlパターンは **BinaryDecimal** となります。

Eclipseのメニューバーより
ファイル→新規→動的Webプロジェクト→「converter」プロジェクトを作成する
→ 図1.1の内容で設定する

図1.1 動的Webプロジェクトの設定

※作成済みであればこの処理は必要ありません

---

urlにアクセスすると図２のような、起動画面が表示されます。ラジオボタンで変換のタイプを選択し、変換したい数値を入力して「変換ボタン」をクリックすると、図３のように変換結果が表示されます。この時、新たな画面へ遷移はしません。

図2．起動画面

図3．変換結果画面

内部設計

クラス連携図は図４のようになります。

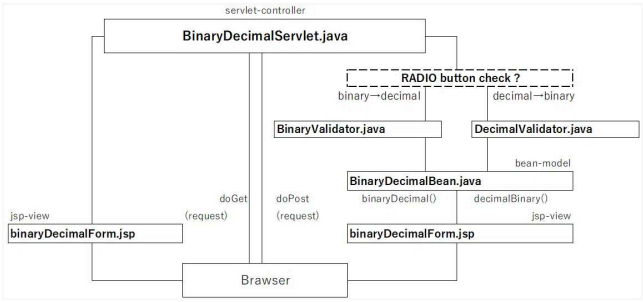

図4．MVCモデル図（クラス連携図）

2進10進変換を行うクラスの仕様とクラス図は以下のようになります。

BinaryDecimalBeanクラス
（仕様）
属性 (フィールド):
・binary: 2進数の値を保持する文字列 (String)
・decimal: 10進数の値を保持する文字列 (String)
コンストラクタ:
・BinaryDecimalBean(): フィールドを初期化するデフォルトコンストラクタ
メソッド:
・binaryDecimal(): binary の値を decimal に変換する
・decimalBinary(): decimal の値を binary に変換する
・setBinary(binary: String): binary 属性を設定する setter メソッド
・setDecimal(decimal: String): decimal 属性を設定する setter メソッド
・getBinary(): String: binary 属性を取得する getter メソッド
・getDecimal(): String: decimal 属性を取得する getter メソッド

（クラス図）

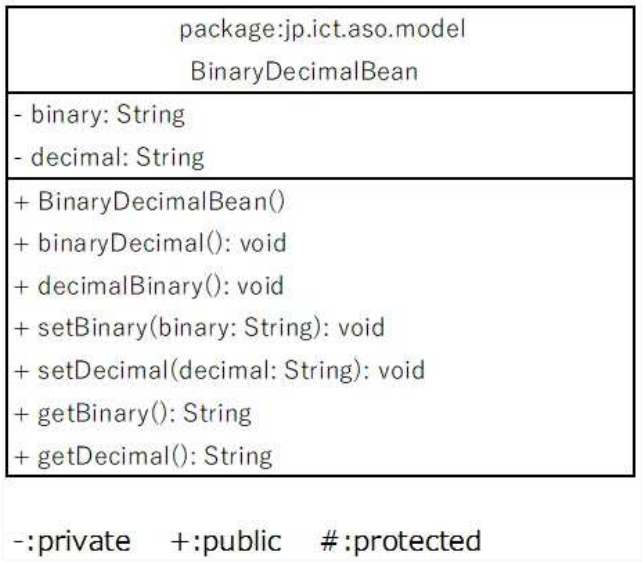

図5．BinaryDecimalBeanクラス図

2進数の範囲チェック、数値チェックを行うクラスの仕様とクラス図は以下のようになります。

BinaryValidatorクラス
（仕様）
メソッド:
入力された文字列が**31ビット以内の2進数**かどうかを判定します。
@param binary 入力された2進数を表す文字列
@return true なら有効な31ビット以内の2進数、false なら無効な値

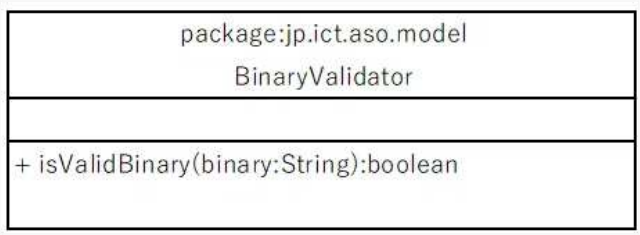

図6．BinaryValidatorクラス図

10進数の範囲チェック、数値チェックを行うクラスの仕様とクラス図は以下のようになります。

DecimalValidatorクラス図
（仕様）
メソッド:
入力された文字列が**-2147483648～2147483647の範囲にある10進数**かどうかを判定します。
@param decimal 入力された10進数を表す文字列
@return true なら有効な範囲の10進数、false なら無効な値

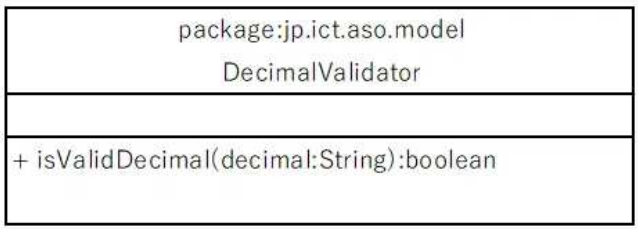

図7．DecimalValidatorクラス図

実装手順

１．２進10進変換ロジック用JavaBeansクラスを作成します

Eclipseパッケージ・エクスプローラより
converterプロジェクトを右クリック→新規→クラス
→以下の内容で作成

・BinaryDecimalBean.java
　　パッケージ：jp.ict.aso.model
　　名前：BinaryDecimalBean
　　ソースコード：考えましょう！（**Swing版と同じです**）

２．２進数入力範囲チェック用クラスを作成します

Eclipseパッケージ・エクスプローラより
converterプロジェクトを右クリック→新規→クラス
→以下の内容で作成

・BinaryValidator.java
　　パッケージ：jp.ict.aso.model
　　名前：BinaryValidator
　　ソースコード：考えましょう！（**Swing版と同じです**）

３．10進数入力範囲チェック用クラスを作成します

Eclipseパッケージ・エクスプローラより
converterプロジェクトを右クリック→新規→クラス
→以下の内容で作成

・DecimalValidator.java
　　パッケージ：jp.ict.aso.model
　　名前：DecimalValidator
　　ソースコード：考えましょう！（**Swing版と同じです**）

４．リクエストコントロール用Servletクラスを作成します

Eclipseパッケージ・エクスプローラより
converterプロジェクトを右クリック→新規→その他→Web→サーブレット
→以下の内容で作成する

・BinaryDecimalServlet.java
　　パッケージ：jp.ict.aso.controller
　　クラス名：BinaryDecimalServlet
　　ソースコード：考えましょう！　※アノテーションは**/BinaryDecimal**
　　(**Swing版のイベントリスナーのコードとほぼ同じです**)

５．起動画面を表示するJSPファイルを作成します

Eclipseパッケージ・エクスプローラより
converterプロジェクトを右クリック→新規→その他→Web→JSPファイル
→以下の内容で作成する

・binaryDecimalForm.jsp
　　保存場所：libcon/src/main/webapp/**WEB-INF/jsp ← 注意！**
　　ファイル名：binaryDecimalForm.jsp
　　ソースコード：考えましょう！

実行確認

サーブレットクラス（BinaryDecimalServlet.java）を実行します！！

Eclipseパッケージ・エクスプローラより
BinaryDecimalServlet..javaを右クリック→実行→サーバーで実行
→図８のように実行される

図8．起動画面

ソースコード例

以下に各プログラムのソースコードの例（本文内では「考えましょう！」になっている部分）を示しますので、実装の参考にしてください。

１．BinaryDecimalBean.java

```
package jp.ict.aso.model;

import java.io.Serializable;

public class BinaryDecimalBean implements Serializable{
    String binary;
    String decimal;
    public BinaryDecimalBean() {
        binary="0";
        decimal="0";
    }
    public void binaryDecimal() {
        //2進10進変換
        int decimalInt = Integer.parseInt(binary, 2);
        decimal=String.valueOf(decimalInt);
    }
    public void decimalBinary() {
        //10進2進変換
```

```
        int decimalInt=Integer.parseInt(decimal);
        binary = Integer.toBinaryString(decimalInt);
    }
    public void setBinary(String binary) {
        this.binary=binary;
    }
    public void setDecimal(String decimal) {
        this.decimal=decimal;
    }
    public String getBinary() {
        return binary;
    }
    public String getDecimal() {
        return decimal;
    }
}
```

2．BinaryValidator.java

```
package jp.ict.aso.model;

public class BinaryValidator {

    /**
     * 入力された文字列が31ビット以内の2進数かどうかを判定します。
     *
     * @param binary 入力された2進数を表す文字列
     * @return true なら有効な31ビット以内の2進数、false なら無効な値
     * intのリテラルの範囲
     *  （int型の変数は -2147483648～2147483647までの数値データを格納）
     */
    public boolean isValidBinary(String binary) {
        // 入力値がnullまたは空文字列でないか確認
        if (binary == null || binary.isEmpty()) {
            return false;
        }

        // 入力値が2進数（0または1）であり、かつ長さが1～31文字であるかをチェック
        return binary.matches("[01]{1,31}");
    }
}
```

3．DecimalValidator.java

```
package jp.ict.aso.model;
```

```
public class DecimalValidator {

    /**
     * 入力された文字列が-2147483648～2147483647の範囲にある10進数かどうかを判定します。
     *
     * @param decimal 入力された10進数を表す文字列
     * @return true なら有効な範囲の10進数、false なら無効な値
     * intのリテラルの範囲
     *  （int型の変数は -2147483648～2147483647までの数値データを格納）
     */
    public boolean isValidDecimal(String decimal) {
        // 入力値がnullまたは空文字列でないか確認
        if (decimal == null || decimal.isEmpty()) {
            return false;
        }

        try {
          // 文字列を整数に変換し、-2147483648～2147483647の範囲に収まっているかをチェック
          int value = Integer.parseInt(decimal);
          return value >= -2147483648 && value <= 2147483647;
               } catch (NumberFormatException e) {
          // 数値への変換が失敗した場合（例：文字列に数字以外の文字が含まれているなど）
          // はfalseを返す
            return false;
        }

    }
}
```

4．BinaryDecimalServlet.java

```
package jp.ict.aso.controller;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import jp.ict.aso.model.BinaryDecimalBean;
import jp.ict.aso.model.BinaryValidator;
import jp.ict.aso.model.DecimalValidator;

/**
 * Servlet implementation class BinaryDecimalServlet
 */
@WebServlet("/BinaryDecimal")
```

```
public class BinaryDecimalServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // メッセージ(なし)をセッションに設定
          String message="";
          HttpSession session=request.getSession();
          session.setAttribute("error", message);
        // 2進↔10進相互変換
          BinaryDecimalBean bd = new BinaryDecimalBean();
        // リクエストスコープに保存
          request.setAttribute("binaryDecimal", bd);
        // フォワード
          RequestDispatcher dispatcher =
            request.getRequestDispatcher
                ("/WEB-INF/jsp/binaryDecimalForm.jsp");
          dispatcher.forward(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // リクエストスコープの取得
          request.setCharacterEncoding("UTF-8");

          String conversionType = request.getParameter("conversionType");
          String numberStr = request.getParameter("number");

          // 変換インスタンス作成
          BinaryDecimalBean bd= new BinaryDecimalBean();
          // 2進数→10進数変換
          if ("binaryToDecimal".equals(conversionType)) {
              //入力値の範囲チェック
                BinaryValidator bv = new BinaryValidator();
                if(bv.isValidBinary(numberStr)) {          //true範囲内
                    // メッセージ(OK)をセッションに設定
                      String message="数値チェックOK";
                      HttpSession session=request.getSession();
                      session.setAttribute("error", message);
                    // 2進数を10進数に変換
                      bd.setBinary(numberStr);
                      bd.binaryDecimal();
                }else {                                    //false範囲外
                     // 数値範囲外ならエラーメッセージをセッションに設定
                      String message="入力された値" + numberStr
                        + " は31ビット以内の2進数ではありません。";
                      HttpSession session=request.getSession();
                      session.setAttribute("error", message);
                }
          // 10進数→2進数変換
          } else if ("decimalToBinary".equals(conversionType)) {
              //入力値の範囲チェック
                DecimalValidator dv = new DecimalValidator();
                if(dv.isValidDecimal(numberStr)) {          //true範囲内
                    // メッセージ(OK)をセッションに設定
                      String message="数値チェックOK";
                      HttpSession session=request.getSession();
                      session.setAttribute("error", message);
                    //10進数を2進数に変換
                      bd.setDecimal(numberStr);
```

```
                    bd.decimalBinary();
                }else {                                          //false範囲外
                    // 数値範囲外ならエラーメッセージをセッションに設定
                    String message="入力された値" + numberStr
                      + " は-2147483648～2147483647の範囲にある10進数ではありません。";
                    HttpSession session=request.getSession();
                    session.setAttribute("error", message);
                }
            } else {
                  // メッセージ(NG)をセッションに設定
                String message="無効な変換タイプです。";
                HttpSession session=request.getSession();
                session.setAttribute("error", message);
            }
            // リクエストスコープに保存
            request.setAttribute("binaryDecimal", bd);

            // フォワード　検索結果出力
            RequestDispatcher dispatcher =
               request.getRequestDispatcher
                   ("/WEB-INF/jsp/binaryDecimalForm.jsp");
               dispatcher.forward(request, response); }

}
```

5．binaryDecimalForm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="jp.ict.aso.model.*" %>
<%
BinaryDecimalBean bd=(BinaryDecimalBean)request.getAttribute("binaryDecimal");
String message=(String)session.getAttribute("error");
//String convert=(String)request.getAttribute("convert");
%>

<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <title>2進10進変換器</title>
</head>
<body>
    <h1>2進数と10進数の相互変換</h1>
    <form action="BinaryDecimal" method="post">
        <input type="radio" name="conversionType" value="binaryToDecimal" required>
        2進数を10進数に変換(31bit以内)
        <br>
        <input type="radio" name="conversionType" value="decimalToBinary" required>
        10進数を2進数に変換(-2147483648～2147483647の範囲)
        <br><br>
        変換する数値:
```

```
        <input type="text" name="number" required><br><br>
        <button type="submit">送信</button>
    </form>
<hr>
<ul>
<li>メッセージ　: <%= message %></li>
<li>  2進数の値　　: <%= bd.getBinary() %></li>
<li>10進数の値　　: <%= bd.getDecimal() %></li>
<!-- <li>コンバート　: <= convert ></li>  -->
</ul>
</body>
</html>
```

連絡先

質問や不明な点がある場合は以下のアドレスにメールをください。24時間以内に返信いたします。

info@slowlife.halfmoon.jp