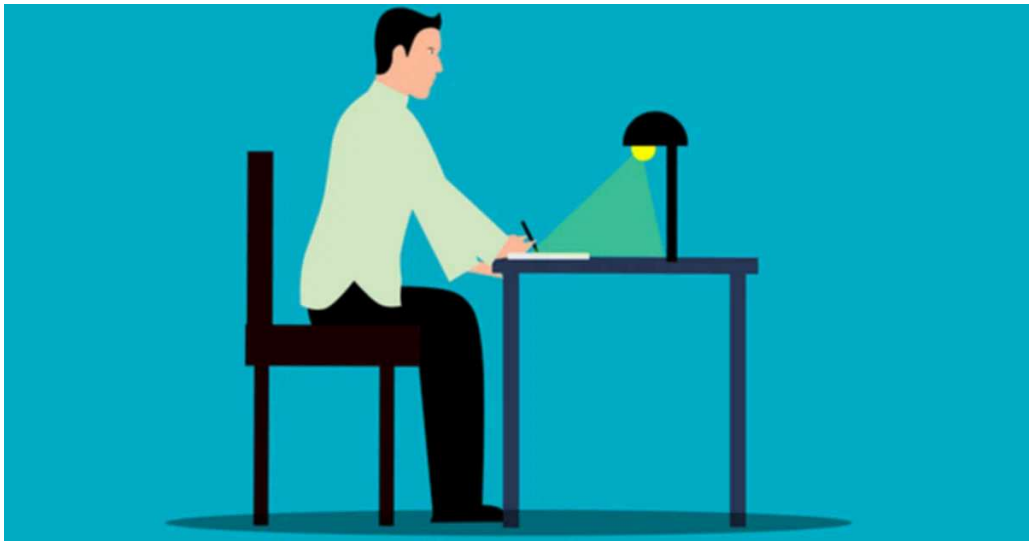

MVCモデルによるWebアプリケーション開発（No９.シーザ暗号）-EE8

office・M
2024年9月3日 12:12

¥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。**今回は部品として作成されたクラスファイルを活用する方法を学びます。**具体的には単一換字式暗号（シーザ暗号）によるエンコード（暗号化）とでデコード（復号）を行うWebアプリを作成します**。暗号化ロジック（Bean）は提供されたクラスを利用します。**チーム開発での役割分担を想定しています。

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。

題材として単一換字式暗号を取り上げます。入力画面に文字列と暗号化キーを入力すると、暗号化（エンコード）された結果を表示します。暗号化を行うBeanは提供します。

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。

今回、業務ロジックであるBeanは作成しません。他のチームメンバーが作成したという想定で提供されたBeanを活用することにします。

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。

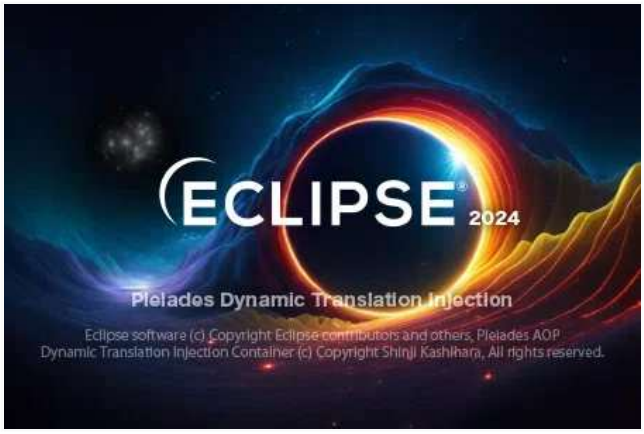

図1．開発環境 Eclipse 2024

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

【単一換字式暗号の仕様】
・暗号化アルゴリズム：特定の文字を辞書順に特定の数だけ前（復号する場合は後ろ）にある文字と置きかえる。この時循環シフトを行う（ローテーションする）。
・暗号化キー：辞書順にずらす数値（特定の数）のこと。とりわけカエサルが実際に用いた３のとき**シーザー暗号**という。

（例）暗号化キーが３の時
　ENCODE（暗号化）　D → A，a → x，2 → 9
　DECODE（復　号）　A → D，x → a，9 → 2

※今回の仕様では大文字、小文字、数字に対応させます（数字に対応させるためkeyの最大値は10とします）

【Model（処理ロジック）の仕様】
CaesarBean.classが提供されます。
・Beanは引数なしのコンストラクタで実体化します。
・ロジックのメソッドを実行することで暗号化(encode)・復号(decode)の

変換処理が行われます。
・結果はフィールド変数(encodeText)(decodeText)内に格納されます。

（実行例）暗号化キーが３の時
INPUT :This is a copy machine of 90 XEROX
ENCODE:Qefp fp x zlmv jxzefkb lc 67 UBOLU
DECODE:This is a copy machine of 90 XEROX

外部設計

接続urlは**http://localhost:8080/caesar/Cipher**とします。よってEclipseの動的Webプロジェクトの名前は**caesar**となり、サーブレットのurlパターンは**Cipher**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→「caesar」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、起動画面を表示します。

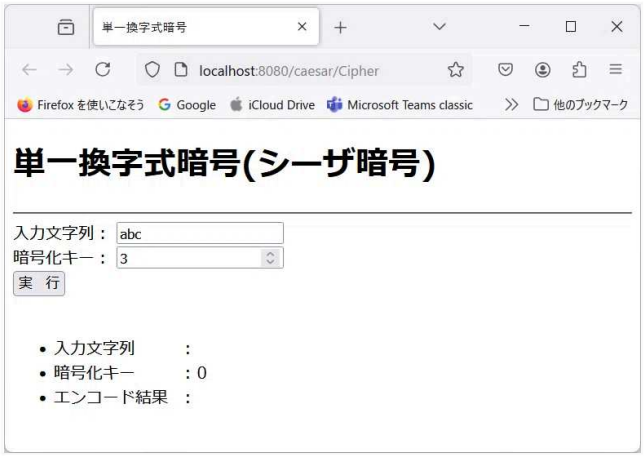

図2．起動画面

入力文字列を入力し暗号化キーを設定して暗号化ボタンをクリックすると、図3のように暗号化された文字列が表示されます。

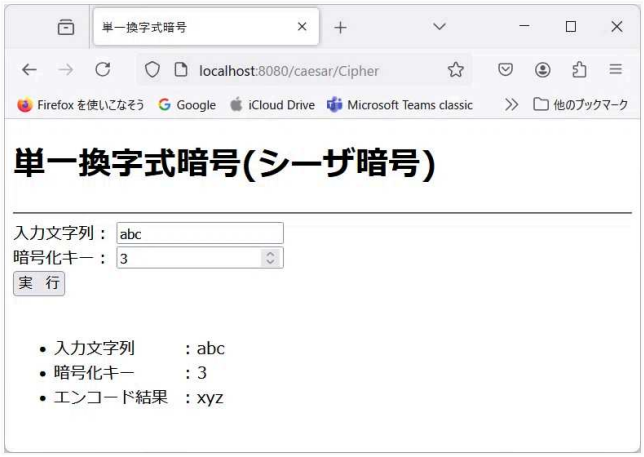

図3．暗号化実行画面

内部設計

クラス連携図は図4のようになります。

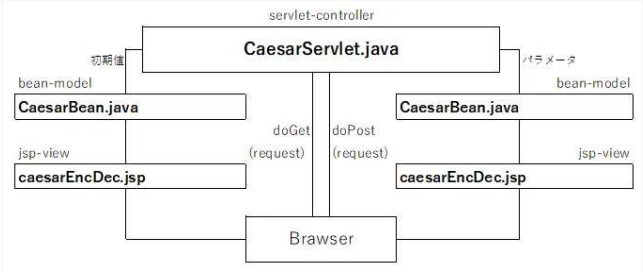

図4．MVCモデル図（クラス連携図）

提供される暗号化・復号計算ロジック（CaesarBean.class）の使い方とクラス図は以下のようになります。

（使い方）

- CaesarBeanは引数なしのコンストラクタで実体化します。
- ロジックのメソッドを実行することで暗号化(encode)・復号(decode)のそれぞれの変換処理が行われます。
- 結果はフィールド変数(encodeText)(decodeText)内にそれぞれ格納されます。

（クラス図）

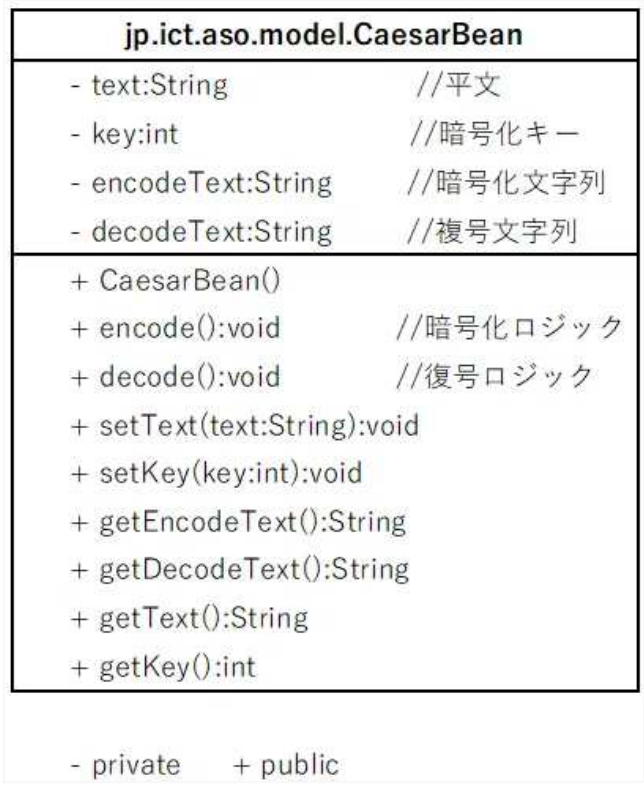

図5．CaesarBeanクラス図

実装手順

１．暗号化・復号計算ロジック用JavaBeansクラスを使用する準備をします

(1)クラスファイルの準備をします

以下Windows環境を想定しています。
事前に「提供Beanフォルダ」を作成しておきます（例　c:￥提供Bean）。
提供Beanフォルダ内に**CaesarBean.class**を保存しておきます。
その際**パッケージの階層**に従ってください。
（例　c:￥提供Bean￥jp￥ict￥aso￥model￥CaesarBean.class）

(2)CaesarBean.classをEclipseのビルド・パスに追加します

Eclipseパッケージ・エクスプローラより
caesarプロジェクトを右クリック→ビルド・パス→ビルド・パスの構成→「ライブラリ」タブ→「クラスパス」クリック→外部クラス・フォルダーの追加→「提供Bean」を指定します→最後に「適用して

閉じる」をクリック

※**Eclipse2024の場合**→caesarプロジェクトを右クリック→プロパティ→Java
　のビルド・パスで設定します。
※図6のように一度設定されていれば再度設定する必要はありません。
※うまく読み込めない場合は一旦「提供Bean」を除去してから再度追
　加するか名前を変えてみます。

図6．Javaのビルドパス追加画面

２．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
caesar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

・CaesarServlet.java
　　パッケージ：jp.ict.aso.controller
　　クラス名：CaesarServlet
　　ソースコード：考えましょう！　※アノテーションは**/Cipher**

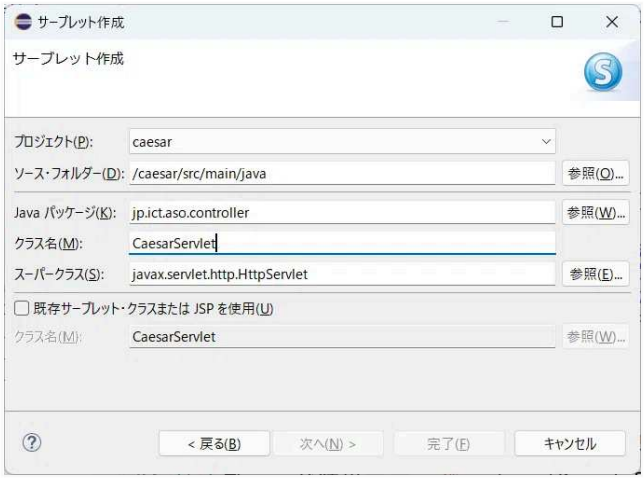

図7．サーブレットの設定

【ヒント】ソースコードの構成：

```
doGest(){
   // CaesarBeanの実体化
   // リクエストスコープに変換結果を保存
   // caesarEncDec.jspにフォワード
}
doPost(){
   // リクエストスコープの取得(caesartext)(caesarkey)
   // CaesarBeanの実体化
   // 文字列セット(caesartext)
   // キーセット(caesarkey)
   // エンコードメソッド実行
   // リクエストスコープに変換結果を保存
   // caesarEncDec.jspにフォワード
}
```

３．入出力画面の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
caesar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

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

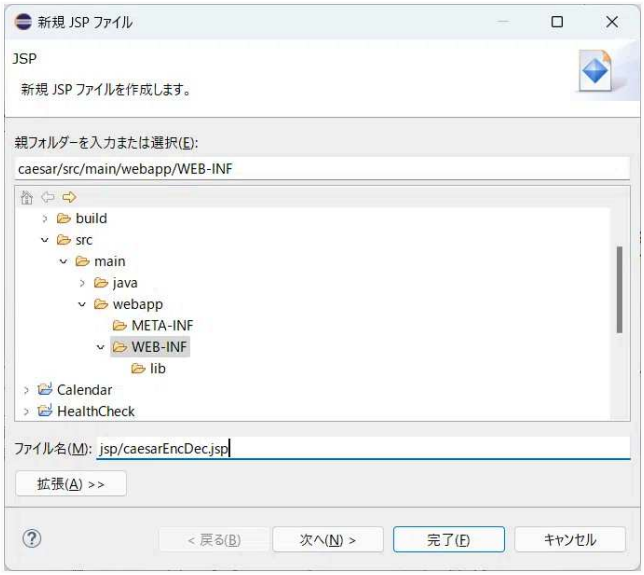

図8．JSPファイルの配置場所

実行確認

サーブレットクラス（CaesarServlet.java）を実行します。しかし、、、

１．InternalServerErrorとなります

実行用Beanのデプロイ（配置）が必要です。この設定を行わないと図9のような実行時エラーになります。

Eclipseパッケージ・エクスプローラより
CaesarServlet.javaを右クリック→実行→サーバーで実行
→図9のように**エラー**となる

図9．実行時エラー500

２．実行用Beanを配置します

CaesarBean.classをEclipseの実行時のクラスパスに追加します。

Eclipse**プロジェクト・エクスプローラ**より
（**パッケージ・エクスプローラではありません！**）
caesarプロジェクトを展開→buildを展開→classesを展開→jpを展開→ictを展開→asoを展開→modelがなければ作成

図10のようにmodelフォルダの位置へCaesarBean.classをドラッグ＆ドロップすることで実行時クラスパスにBeanが追加されます。
※Eclipseを終了させると、再度追加が必要な場合があります。
※作成したはずのフォルダが表示されない、model位置に当該クラスが表示
　されないときは**F5**でリフレッシュしてください

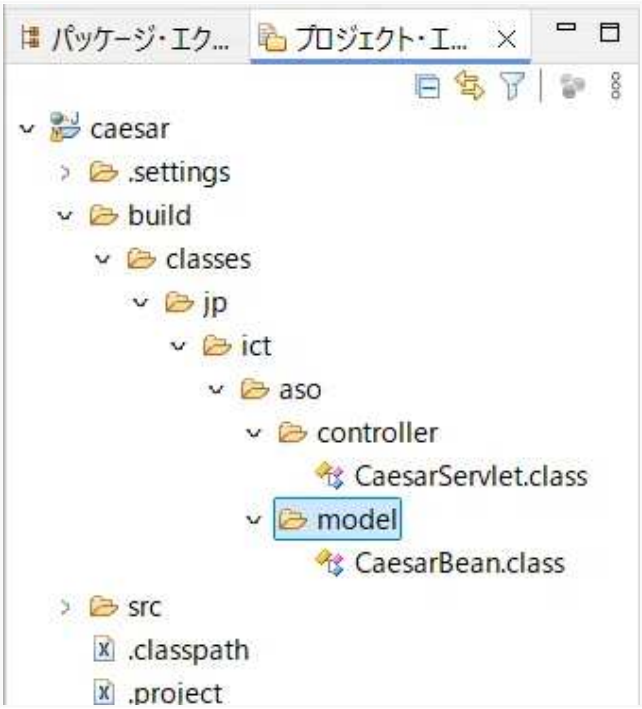

図10．実行時クラスパスの位置

３．再度実行します

Tomcatサーバを再起動したのち、再度サーブレットクラス（CaesarServlet.java）を実行します。

Eclipseパッケージ・エクスプローラより
CaesarServlet.javaを右クリック→実行→サーバーで実行
→図11のように実行される

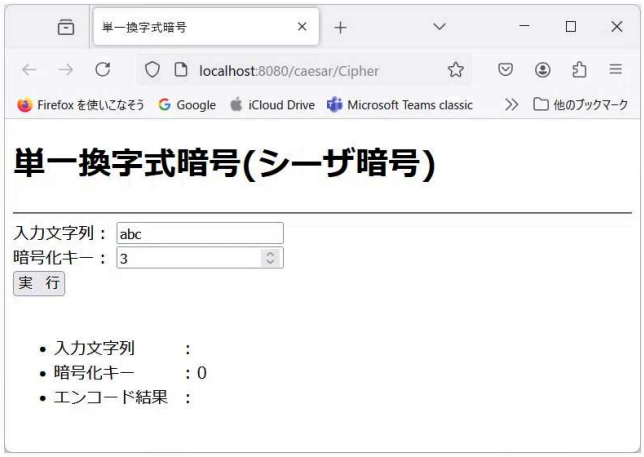

図11．起動画面

ソースコード例と提供Bean

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。

また、提供BeanとしてCaesarBean.classを次のセクションに置きますので、ダウンロードして使用してください。

このラインより上のエリアが無料で表示されます。

1．CaesarBean.class

CaesarBean.class

2.67 KB

ファイルダウンロードについて

ダウンロード

2．CaesarServlet.java

```
1  package jp.ict.aso.controller;
2
3* import java.io.IOException;
13
14 @WebServlet("/Cipher")
15 public class CaesarServlet extends HttpServlet {
16
17*   protected void doGet(HttpServletRequest request,
18        HttpServletResponse response)
19        throws ServletException, IOException {
20
21      // エンコード・デコード
22        CaesarBean cb = new CaesarBean();
23      // リクエストスコープに保存
24        request.setAttribute("caesar", cb);
25
26        // フォワード
27        RequestDispatcher dispatcher =
28          request.getRequestDispatcher
29              ("/WEB-INF/jsp/caesarEncDec.jsp");
30        dispatcher.forward(request, response);
31    }
```

図12．CaesarServlet.java（１）

```java
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 caesarText = request.getParameter("caesartext");
    String strCaesarKey = request.getParameter("caesarkey");
    int caesarKey=Integer.parseInt(strCaesarKey);

  // エンコード
    CaesarBean cb = new CaesarBean();
    cb.setText(caesarText);
    cb.setKey(caesarKey);
    cb.encode();


    // リクエストスコープに保存
    request.setAttribute("caesar", cb);

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

図13．CaesarServlet.java（２）

３．caesarEncDec.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="jp.ict.aso.model.*" %>
<%
CaesarBean cb=(CaesarBean)request.getAttribute("caesar");
%>
<!DOCTYPE html>
<html>
<head>
<title>単一換字式暗号</title>
</head>
<body>
<h1>単一換字式暗号(シーザ暗号)</h1>
<hr>
        <form method="POST" action="/caesar/Cipher">
            入力文字列： <input type="text" name="caesartext" value="abc"> <br>
            暗号化キー： <input type="number" name="caesarkey" value="3"> <br>
            <input type="submit" value="実　行">
        </form><br>

<ul>
<li>入力文字列　　　： <%= cb.getText() %></li>
<li>暗号化キー　　　： <%= cb.getKey() %></li>
<li>エンコード結果　： <%= cb.getEncodeText() %></li>

</ul>

</body>
</html>
```

図14．caesarEncDec.jsp

問題

①図15のような復号を行う機能を追加してください。

ヒント：変更するのはCaesarServlet.javaとcaesarEncDec.jspで１行追加するだけです。

②以下の暗号文を解読し、その内容の答えを報告してください。

OZSL AK LZW DGFYWKL JANWJ AF BSHSF

ヒント：シーザ暗号ではありません（暗号化キーは３ではありません）。

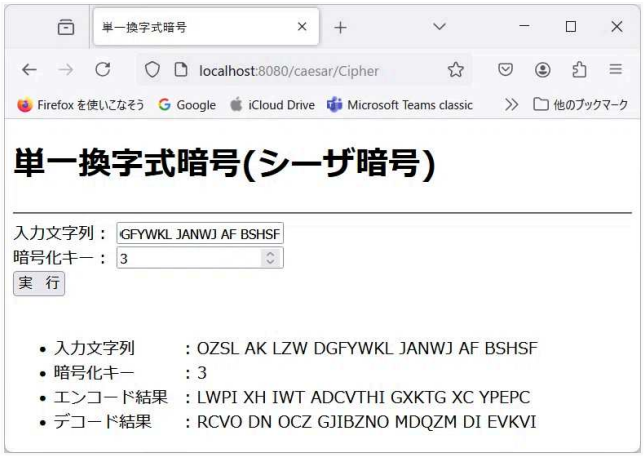

図15．復号（デコード）機能の追加

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