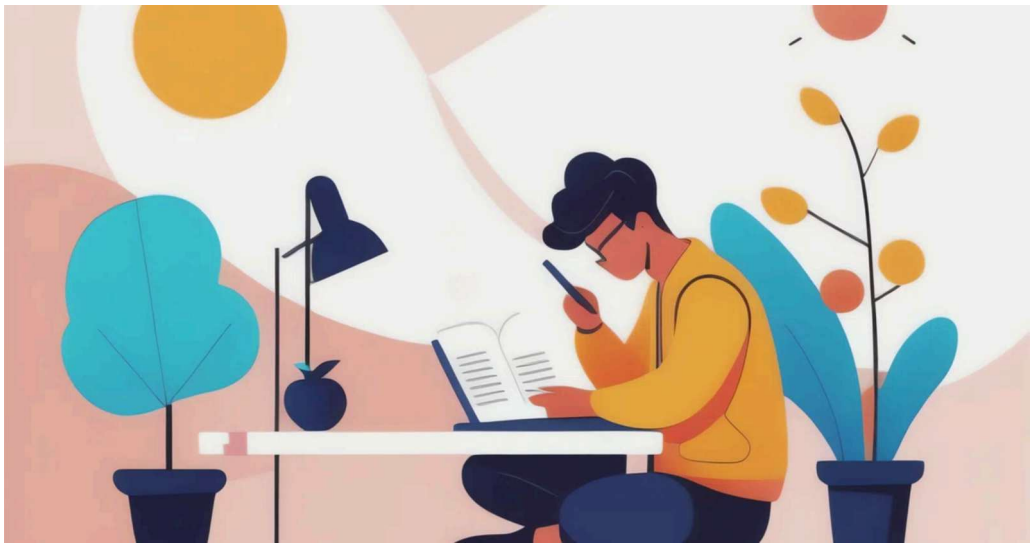


MVCモデルによるWebアプリケーション開発（No７.貸付返済）-EE8

office・M
2024年1月12日 13:56

¥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（リボルビング型カードローンの返済シミュレーションをhtml形式で出力するBean）を活用する方法を学びます。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

4．シミュレーション結果の出力画面のJSPファイルを作成します

実行確認

1．InternalServerErrorとなります

2．実行用Beanを配置します

3．再度実行します

ソースコード例と提供Bean

1．LoanSimBean.class

2．LoanServlet.java

3．loanForm.jsp

4．loanResult.jsp

連絡先

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。

題材としてリボルビング型カードローンの返済シミュレーションを行うWebアプリケーションを作成します。リボルビング型ローンなので毎月の返済額が一定となり、借入残額によって返済期間が変動します。入力画面に借入額、利率、返済額を入力すると、返済期間のシミュレーション結果を表示する画面に遷移します。返済期間を計算するBeanは提供します。このBeanは、参照先に示す「アプラス（新生銀行）　元利定額リボルビング払いシミュレーション」と同じ結果を返すようオリジナルで作成したものです。

（参照先）
アプラス（新生銀行）　元利定額リボルビング払いシミュレーション
https://www.aplus.co.jp/creditcard/revo/simulation/index.html

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

【貸付返済シミュレーションの仕様】
以下の条件を実現するWebアプリケーションの作成を行います。
（条件）
・urlを指定してサイトにアクセスすると３つの入力欄（借入額、利率、返済額）と「シミュレーションボタン」が配置された画面を表示します。
・借入額とは借入残金、利率とは実質年率、返済額とは月々の返済金額を表しています。
・借入額は円単位で、利率は%の実数値を小数点以下2桁で、返済額は円単位で入力します。
・貸付利子（手数料ともいいます）の計算は「借入残金×(年率÷12)」で端数切捨てで求めています。
・返済額は一定です。つまり返済額＝借入残金充当額＋貸付利子となります。
・「シミュレーションボタン」をクリックすると画面が遷移して月毎の借入残金の履歴を表示します。

【Model（処理ロジック）】
LoanSimBean.classが提供されます。

（仕様）
・引数で借入額(円)と年利率(実数値)と返済金額(円)を受け取る
・月毎の借入残金を計算しhtml形式で変数に保持する
・借入残金の変数はメソッドで外部に公開される

外部設計

接続urlは**http://localhost:8080/money/Loan**とします。よってEclipseの動的Webプロジェクトの名前は**money**となり、サーブレットのurlパターンは**Loan**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→「money」プロジェクトを作成する→図1.1の内容で設定する

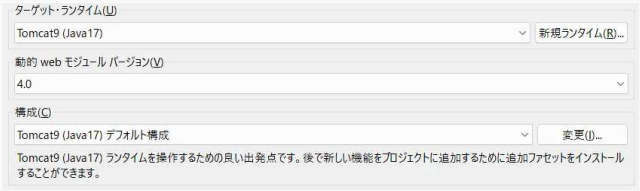


図1.1 動的Webプロジェクトの設定

※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。借入額(初期借入額)と年率(想定利回り)と期間(積立総月数)を入力して「シミュレーションボタン」をクリックすると、図３のように借入残金のシミュレーション結果の表示画面に遷移します。

図２．起動画面

図３．借入残金のシミュレーション画面

最終的には「マネーシミュレーションWebアプリ」のメニュー画面からクリッカブルマップを利用して起動できるように構成します。

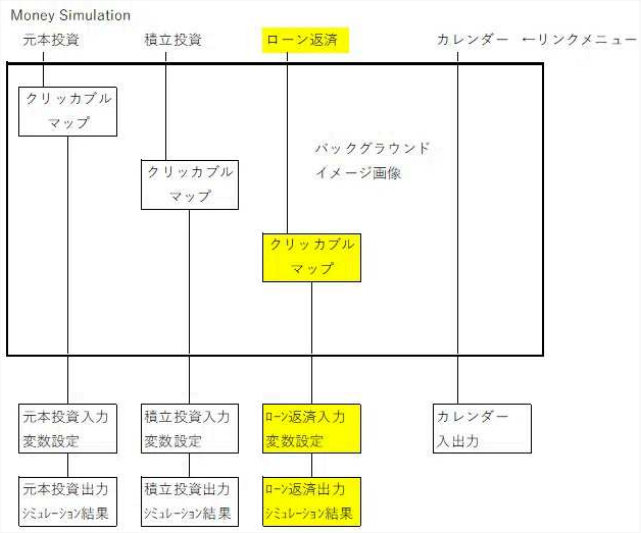


図3.1 マネーシミュレーションWebアプリ概念図

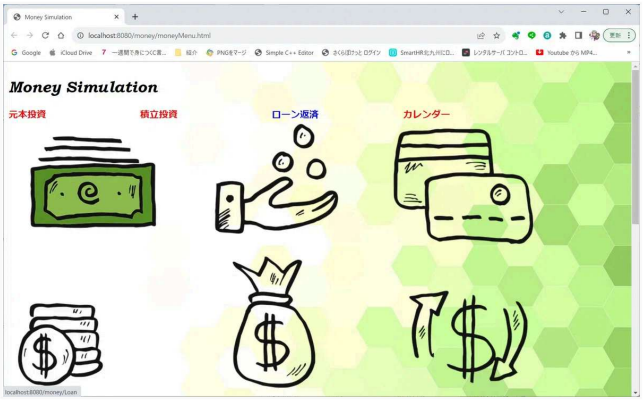


図3.2 マネーシミュレーションWebアプリの例

内部設計

クラス連携図は図４のようになります。

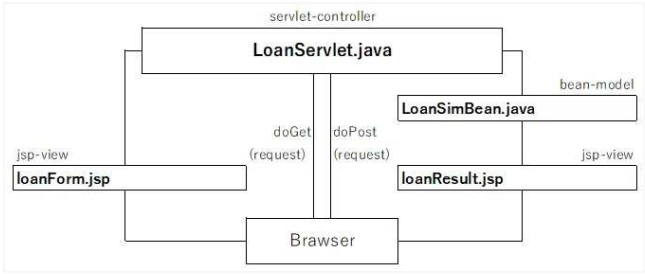


図４．MVCモデル図

提供される貸付返済計算ロジック（LoanSimBean.class）の使い方とクラス図は以下のようになります。

（使い方）
・LoanSimBeanクラスを**借入残額**と**年利率**と**返済金額**の３つの引数をもつコンストラクタで実体化します。
・simulationメソッドを実行することでsimフィールドに借入残金の履歴がhtml文字列で表組されます。
・利用側のクラスはgetSim()メソッドで借入残金の履歴を取得します。

（クラス図）

図4.1 LoanSimBeanクラス図

１．貸付返済計算ロジック用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**LoanSimBean.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￥LoanSimBean.class）

(2)LoanSimBean.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
money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

※図５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。

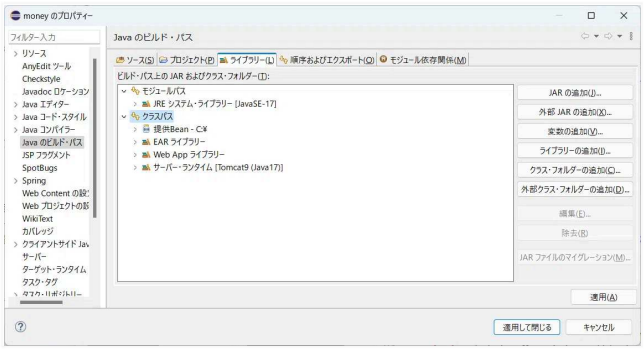


図５．Javaのビルドパス追加画面

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
money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

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

３．借入額・利率・返済額の入力画面の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
money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

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

４．シミュレーション結果の出力画面の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
money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

・loanResult.jsp
　保存場所：money/src/main/webapp/**WEB-INF/jsp ← 注意！**
　ファイル名：loanResult.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（LoanServlet.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６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
LoanServlet.javaを右クリック→実行→サーバーで実行

→図６のように**エラー**となる

図６．実行時エラー画面

２．実行用Beanを配置します

LoanSimBean.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！**）
money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７のようにmodelフォルダの位置へLoanSimBean.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。

図７．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（LoanServlet.java）を実行します。

Eclipseパッケージ・エクスプローラより
LoanServlet.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

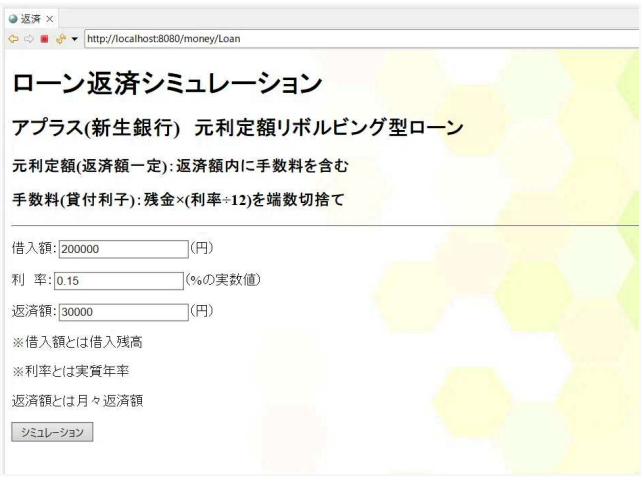


図８．起動画面

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LoanSImBean.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。

１．LoanSimBean.class

LoanSimBean.class

2.71 KB

ダウンロード

ファイルダウンロードについて

2．LoanServlet.java

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

import java.io.IOException;

@WebServlet("/Loan")
public class LoanServlet extends HttpServlet {

  protected void doGet(HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException {

    // フォワード
      RequestDispatcher dispatcher =
        request.getRequestDispatcher
            ("/WEB-INF/jsp/loanForm.jsp");
      dispatcher.forward(request, response);
  }
  protected void doPost(HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException {

    // リクエストパラメータの取得
    request.setCharacterEncoding("UTF-8");
    String szankin = request.getParameter("zankin");
    String sriritu = request.getParameter("ririru");
    String shensai = request.getParameter("hensai");

    int zankin = Integer.parseInt(szankin);
    double ririru = Double.parseDouble(sriritu);
    int hensai = Integer.parseInt(shensai);
```

図９．LoanServlet.java（１）

```java

    // 入力値をプロパティに設定
    LoanSimBean lsb = new LoanSimBean(zankin,ririru,hensai);
    lsb.simulation();

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

    // フォワード
    RequestDispatcher dispatcher =
        request.getRequestDispatcher
            ("/WEB-INF/jsp/loanResult.jsp");
    dispatcher.forward(request, response);
  }
}
```

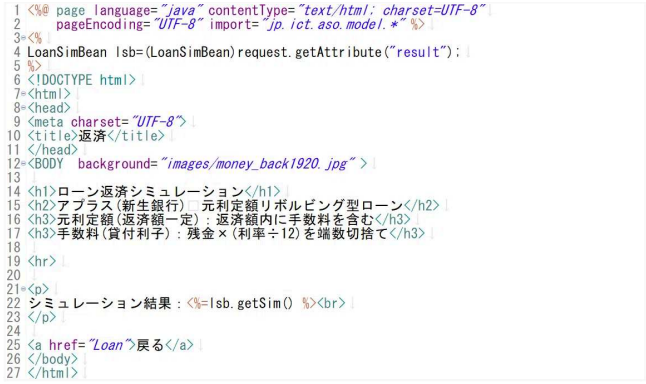
図10．LoanServlet.java（２）

3．loanForm.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>返済</title>
</head>
<BODY background="images/money_back1920.jpg">

<h1>ローン返済シミュレーション</h1>
<h2>アプラス（新生銀行） 元利定額リボルビング型ローン</h2>
<h3>元利定額（返済額一定）：返済額内に手数料を含む</h3>
<h3>手数料（貸付利子）：残金×（利率÷12）を端数切捨て</h3>
<hr>

        <form method="POST" action="Loan">
<p>借入額：<input type="number" name="zankin" value=200000> （円） </p>
<p>利　率：<input type="number" name="ririru" step=0.001 value=0.15> （%の実数値） </p>
<p>返済額：<input type="number" name="hensai" value=30000> （円） </p>
<p>※借入額とは借入残高</p><p>※利率とは実質年率</p>
<p>返済額とは月々返済額</p>

<input type="submit" value="シミュレーション">

        </form>

</body>
</html>
```

図11．loanForm.jsp

4．loanResult.jsp



```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="jp.ict.aso.model.*" %>
<%
LoanSimBean lsb=(LoanSimBean)request.getAttribute("result");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>返済</title>
</head>
<BODY  background="images/money_back1920.jpg" >

<h1>ローン返済シミュレーション</h1>
<h2>アプラス(新生銀行)　元利定額リボルビング型ローン</h2>
<h3>元利定額(返済額一定)：返済額内に手数料を含む</h3>
<h3>手数料(貸付利子)：残金×(利率÷12)を端数切捨て</h3>

<hr>

<p>
シミュレーション結果：<%=lsb.getSim() %><br>
</p>

<a href="Loan">戻る</a>
</body>
</html>
```

図12．loanResult.jsp

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