最速でJavaエンジニアになって稼ぐ

2018/11/10

人生逃げ切りオンラインサロン内

目次的なやつ

- CRUD処理
- DB(データベース)とSQL
- ドメインモデル
- CRUD処理をWebアプリケーションとして作ってみる
- 様々な外部ライブラリと構成管理
- SpringBoot + Thymeleaf

CRUD処理

- CRUD処理とは「Create(生成/登録)」「Read(読取)」「Update(更新)」「Delete(削除)」の頭文字を取ったもの。
- 業務系のシステムにおいては「マスターメンテナンス」と呼ばれるものにこの処理が入る。
- このCRUD処理はアプリケーション自体の振る舞いとしても考えることが出来ると同時に、次で話す「DB(データベース)」が持つ主要機能としての振る舞いとしても考えられる。
- いわゆる「アプリケーションの基本形」みたいなもの(テンプレ的な)。
- 機能毎に全てCRUDを入れるわけではなく、Readのみ等も勿論ある。
- 逆にここから逸脱することもないので、CRUD処理が作れるなら、後は詳細がどうなるかだけで殆どのアプリケーションは構成可能。

DB(データベース)とSQL

DBとはデータを整理して検索しやすくした情報の集まりの事

- 現在のIT分野で利用されるDBの殆どは「RDB(リレーショナルデータベース)」と呼ばれ、詳細は割愛するけど、データの関係性に基づいて設計や定義されるDBの事。
- 更にそのRDBを管理する為のシステムがあり、略して「RDBMS」と呼ばれる。
- 「OracleDataBase」「MySQL」「SQL Server」「DB2」「PostgreSQL」「H2 Database」などが代表的なRDBMSで現場によって使うものは違う。
- Javaの現場においては殆どが「Oracle」か「MySQL」になる。これは開発規模やサポート体制などの選定基準によるものと、昔からOracleがDBとしては強かったためにそのまま、というところもある。
- それぞれのDBには特徴があり、また開発された際の思想も違う。利用できるSQLが違うといったものもあり、良くも悪くも一長一短で幅広い。

SQLとは、DBに対してデータの操作を
行う為の「言語」

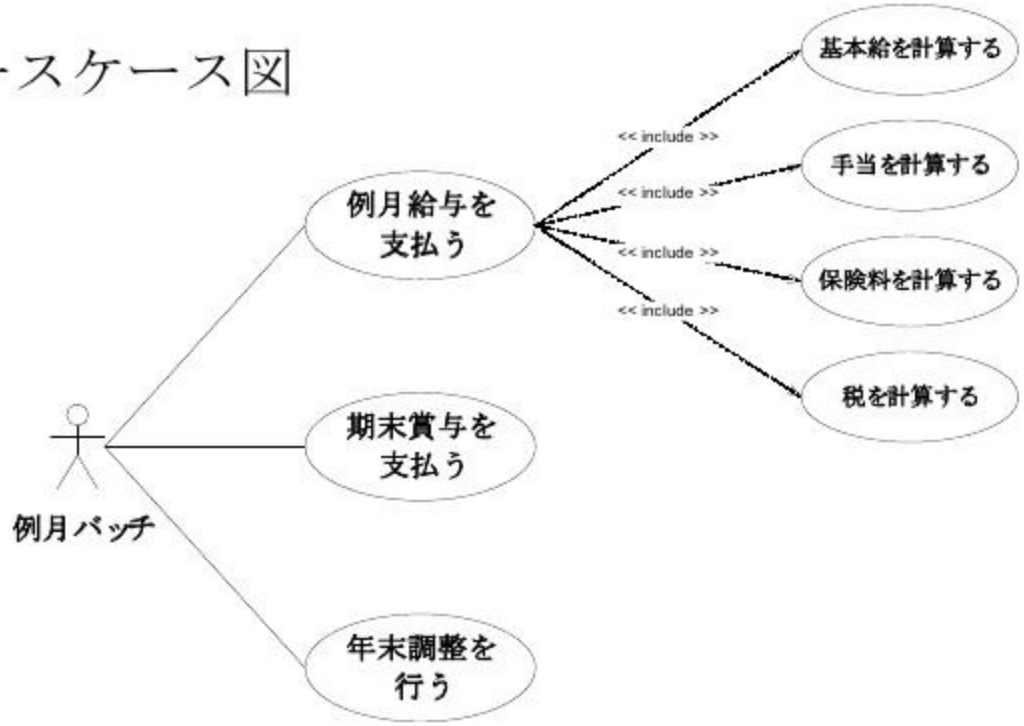
- DB自体がデータを集積するモノである以上、それを取り出したり、追加したりする事も勿論可能。それを行うのに利用するのが「SQL」
- アプリケーションの基本形として紹介したCRUD処理は、DBにも適用可能。
- ただし「データの操作」を行うのが「SQL」なので、CRUD以外も勿論可能。
- ちなみにDBに対するCRUDに当たる部分はSQLで使う言葉に合わせて「INSERT」「SELECT」「UPDATE」「DELETE」になる。
- アプリケーションやプログラムで扱う殆どが「データ」なので、そのデータをどう取り扱うかを決めるDBとSQLの組み合わせはアプリを作る上で必須技術。
- ちなみに、SQLだけでプログラムと同等の処理を実現出来るので、極めるとSQLで色々出来てしまう。(それが良いかどうかはまた別)
- 蛇足。SQLを発行するという事を「クエリを投げる(発行する)」と言ったりする。

ドメインモデル

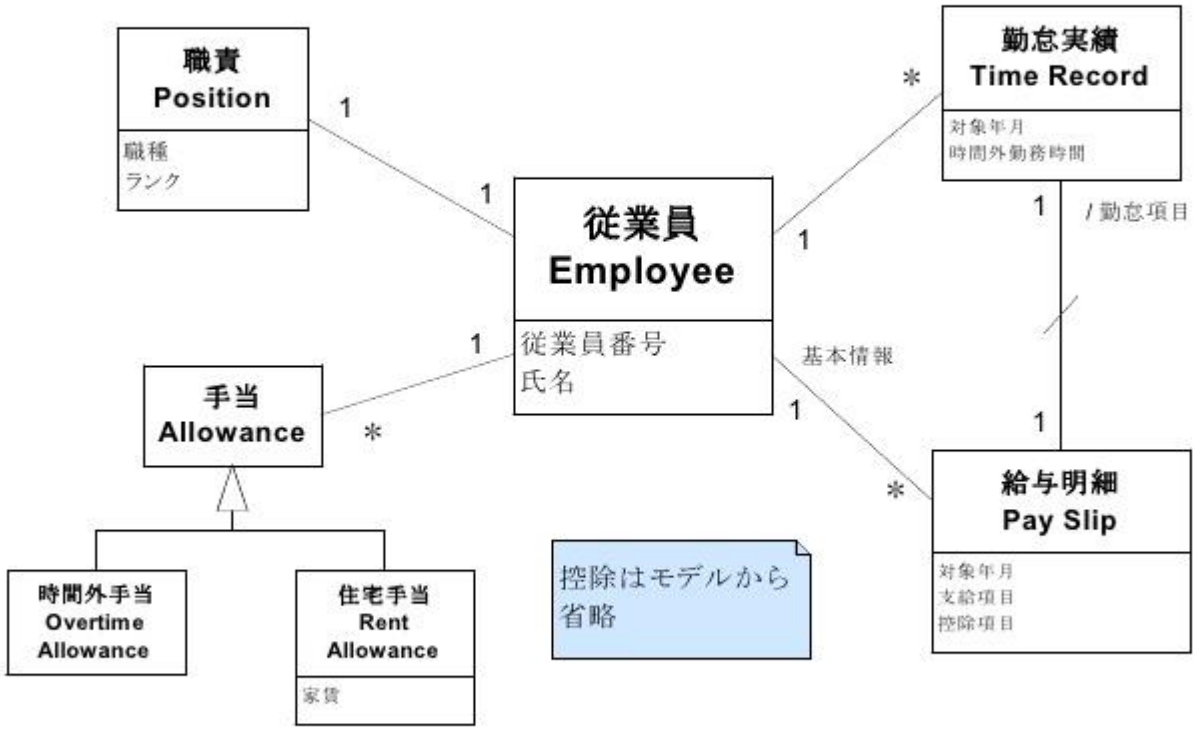
アプリケーション(システム)を構築する際の領域(ドメイン)の関係性を表現(モデリング)したモノを「ドメインモデル」と呼ぶ。

給与計算システムの例

ユースケース図



給与システムのドメインモデル



正直、最初は理解してなくてもOK。
だってドメインモデリングをするような仕事はかなり経験を踏んでからだから。

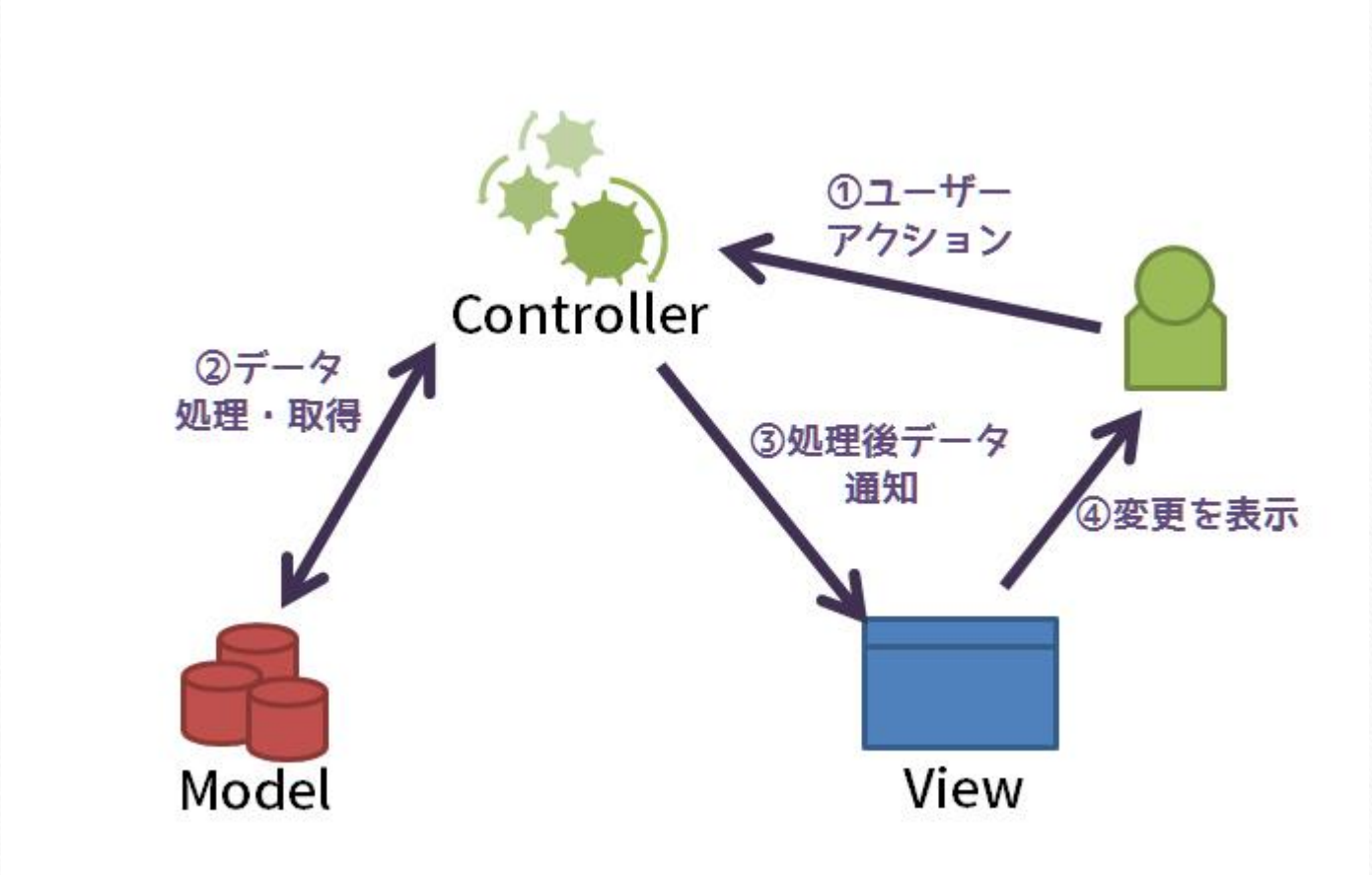
MVCモデルやMVVMモデル

- MVC(Model/View/Controller)に分割してプログラムを書くモデル。
- MVVM(Model/View/ViewModel)に分割して(ry

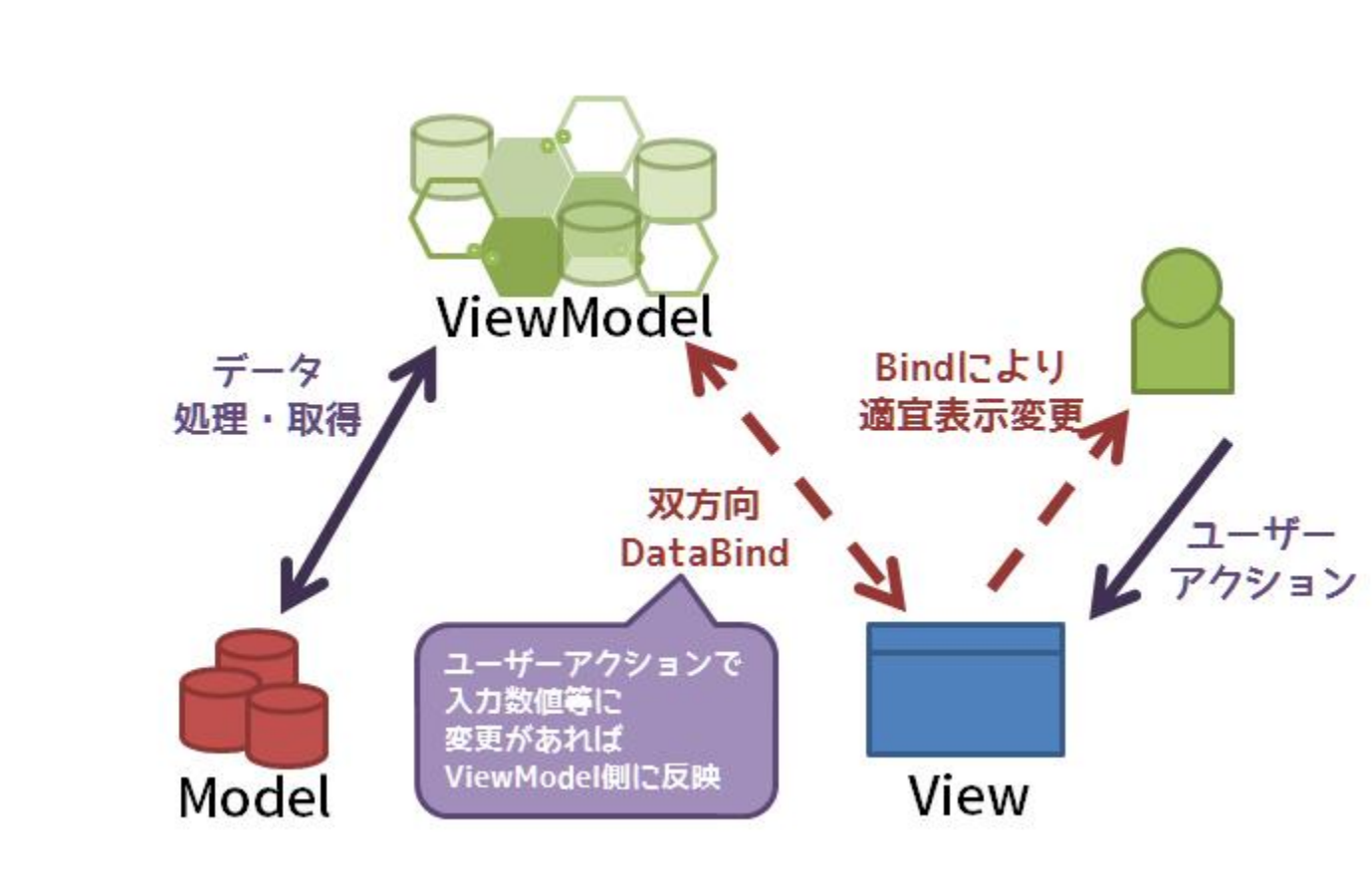
- Model　→　ビジネスロジック(業務処理)を書く場所
- View　→　表示とか入出力について書く場所。自動描画の場合も有。
- Controller　→　ユーザー処理(リクエスト)によってMとVを制御する場所
- ViewModel　→　MとVの伝達役。Viewの為の状態を保持する場所。

わけわからんので図解。

MVCモデル


①ユーザー
アクション
Controller
②データ
処理・取得
③処理後データ
通知
④変更を表示
Model
View

MVVMモデル


ViewModel
データ
処理・取得
Bindにより
適宜表示変更
双方向
DataBind
ユーザー
アクション
ユーザーアクションで
入力数値等に
変更があれば
ViewModel側に反映
Model
View

とりあえずは、MVCモデルだけ押さえておいてくれれば問題なし。

CRUD処理をWebアプリケーションとして作ってみる

MySQLのインストールと初期設定
https://webkaru.net/mysql/install-windows/
https://webkaru.net/mysql/windows-confirmation/

テーブルの作り方
https://techacademy.jp/magazine/5113

JDBCドライバーのインストールとCRUD処理のサンプル
https://www.task-notes.com/entry/20150414/1428980400

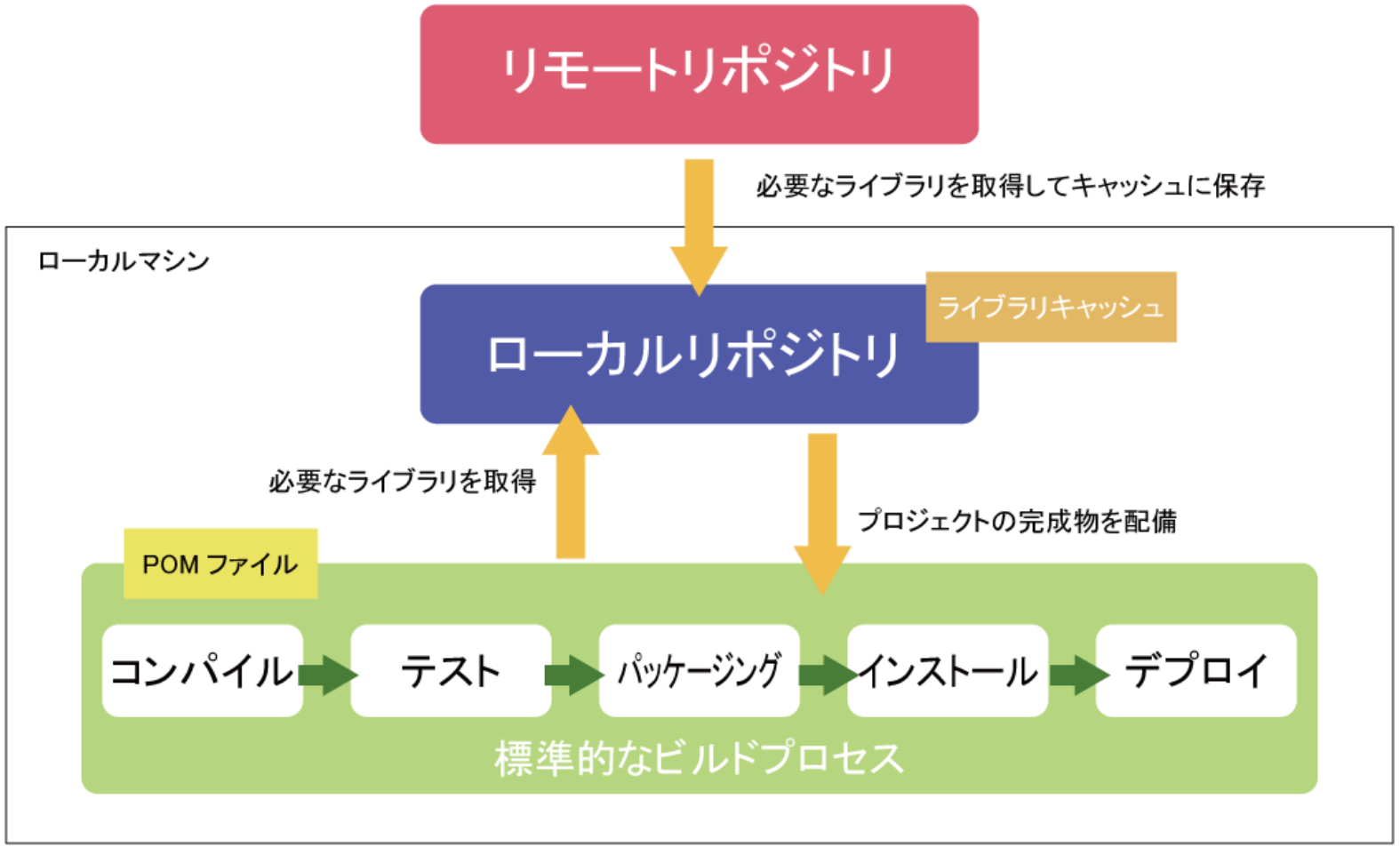
様々な外部ライブラリと構成管理

外部ライブラリは「JDBCドライバー」のような「他の誰か」が作った「便利機能」の集まりの事。

- 「こんな機能ないかな」と思った時点で、大体他の誰かが「既に作ってる」のが殆どなので、まずは探す。
- 見つけたらそれを今まで追加してきたような感じで「外部jar」として追加すれば、自分のアプリケーションでも使える。
- 勿論、ちゃんと公開されていて「使ってもいいよ」と言われているものに限るけど、殆どが使える、かつ、カスタマイズ可能。
- SpringやStrutsなどのフレームワークも言ってしまえば、そういった便利機能を集めまくって構成されたものと言える。一部分だけSpringの機能を使う、というような事も可能。
- ライブラリとして有名なのは「Apache Commons」と呼ばれるもの。プログラムをする上で欲しくなるような機能が大体揃っている。

外部jarとして毎回追加するの、めんどい。

構成管理ツールを使おう！


リモートリポジトリ
必要なライブラリを取得してキャッシュに保存
ローカルマシン
ローカルリポジトリ
ライブラリキャッシュ
必要なライブラリを取得
プロジェクトの完成物を配備
POM ファイル
コンパイル
テスト
パッケージング
インストール
デプロイ
標準的なビルドプロセス

build.gradleの中身のサンプル

```
buildscript {
    ext {
        springBootVersion = '2.1.0.RELEASE'
    }
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}")
    }
}

apply plugin: 'java'
apply plugin: 'eclipse-wtp'
apply plugin: 'org.springframework.boot'
apply plugin: 'io.spring.dependency-management'
apply plugin: 'war'

group = 'study.app'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

configurations {
    providedRuntime
}

dependencies {
    implementation('org.springframework.boot:spring-boot-starter-thymeleaf')
    implementation('org.springframework.boot:spring-boot-starter-web')
    providedRuntime('org.springframework.boot:spring-boot-starter-tomcat')
    testImplementation('org.springframework.boot:spring-boot-starter-test')
}
```

- 構成管理ツールを活用することで、そのアプリケーションに必要な「ライブラリや構成」を簡単に作成する事が出来る。
- 外部ライブラリをいちいち追加する必要はなく、構成管理として関係性を定義(依存関係を定義と言う)しておけば、自動的に外部ライブラリとして扱える。
- リモートリポジトリに当たるものは「MavenCentral」という名前で公開されており、自由に使え、殆どのモジュールやフレームワークもここを通して利用する。
- 今どきのJava開発で「外部jarを追加」とか存在しない。
- 慣れてくるとアプリケーション開発をするのに必要なモジュール構成を一瞬で用意出来るようになるので、お手軽感が増す。

SpringBoot + Thymeleaf

SpringBootとはSpringベースのアプリケーションを簡単に作成できるようにするためのフレームワーク

極端な言い方をすれば
「Java版のRuby on Rails」

従来のJava開発(Struts含む)と
SpringBootでは別物と思っても良いぐらい違う。

ThymeleafとはHTML形式で用意された「テンプレート」を動的に書き換える事によって、画面描画を行う「テンプレートエンジン」

こんな感じ。

home.html

```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-thymeleaf-4.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">

  <head>
    <title>Good Thymes Virtual Grocery</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" type="text/css" media="all"
          href="../../css/gtvg.css" th:href="@{/css/gtvg.css}" />
  </head>

  <body>

    <p th:text="#{home.welcome}">Welcome to our grocery store!</p>

  </body>

</html>
```

JSPなんてなかった。

実際のところ、JSPを使う現場はそれなりに残っているけれど、流れとしてはテンプレートエンジンを利用する方向。

とはいえSpringBootもThymeleafも
使えなきゃ意味なし。

なのでこれからは

SpringBoot + Thymeleaf + 構成管理ツール(Gradle)

でやっていきます。(追加もあるよ

SpringBootの起動まで
https://www.marineroad.com/staff-blog/16785.html

SpringBoot + Thymeleafの動作確認まで(↑の続き)
https://www.marineroad.com/staff-blog/17070.html

STSのインストールやSpringBootの説明捕捉
https://eng-entrance.com/java-springboot

次回までの宿題

出来なかった人は出来るまで。
出来た人は今回のSpringBootアプリで
CRUD処理を実装してみよう。

おしまい