spring-sample

Spring bootのサンプルプロジェクトです。
RESTFulなAPIでデータベースにアクセスしてデータの更新、取得が行えます。

前提条件

- eclipse にLombokが導入済み
- java8以上がインストール済み
- postgresqlがインストール済み

LombaokについてはpleiadesのLombok付きをインストールすると楽pleiades

動作確認環境

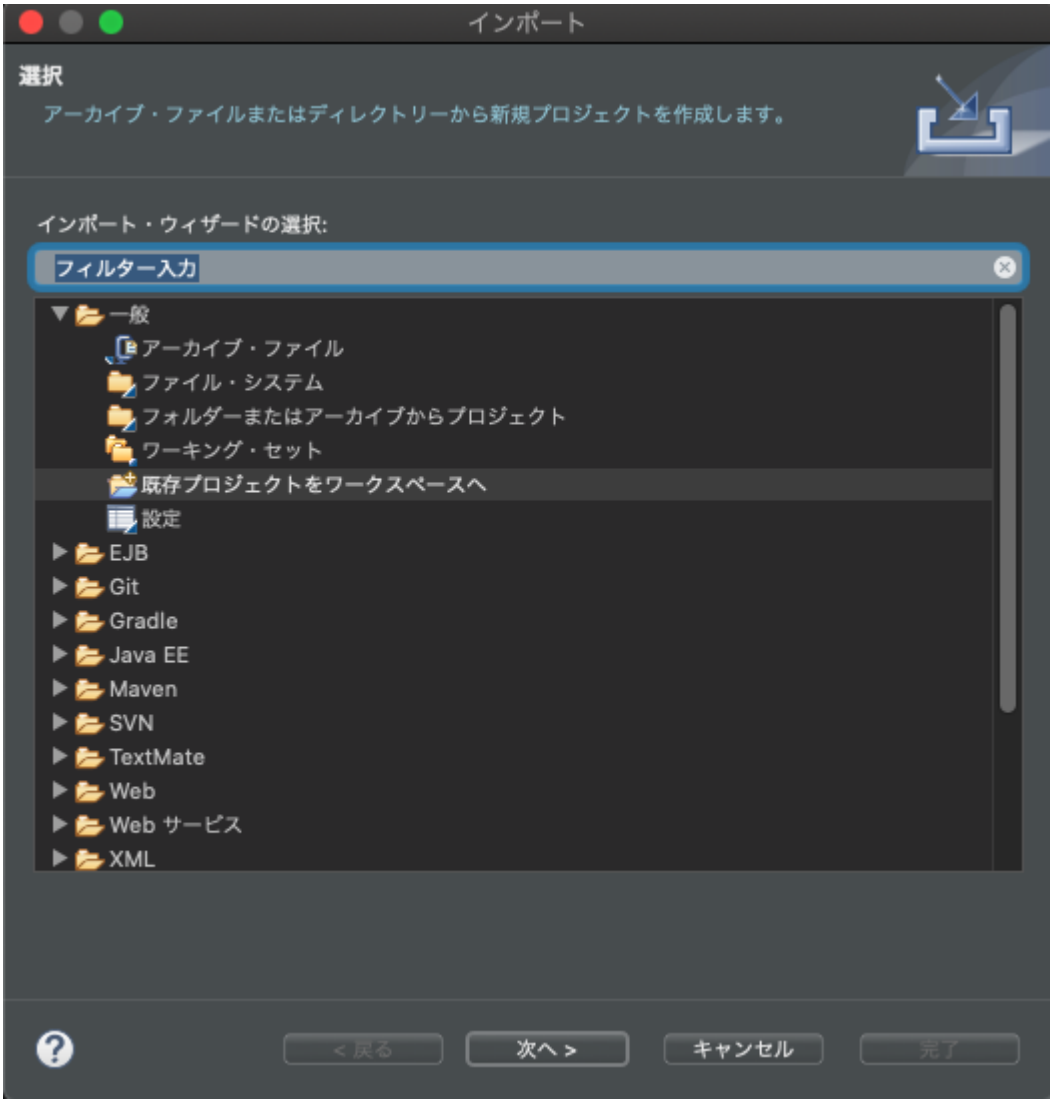
環境	サービス/バージョン
実行環境	Mac OS Catalina、Windows10
開発環境	eclipse pleiades 2019
開発言語	Java 8
DB	PostgreSQL 10
Framework	SpringBoot 2.2.6
依存関係	Jersey、MyBatis、Jackson

手順

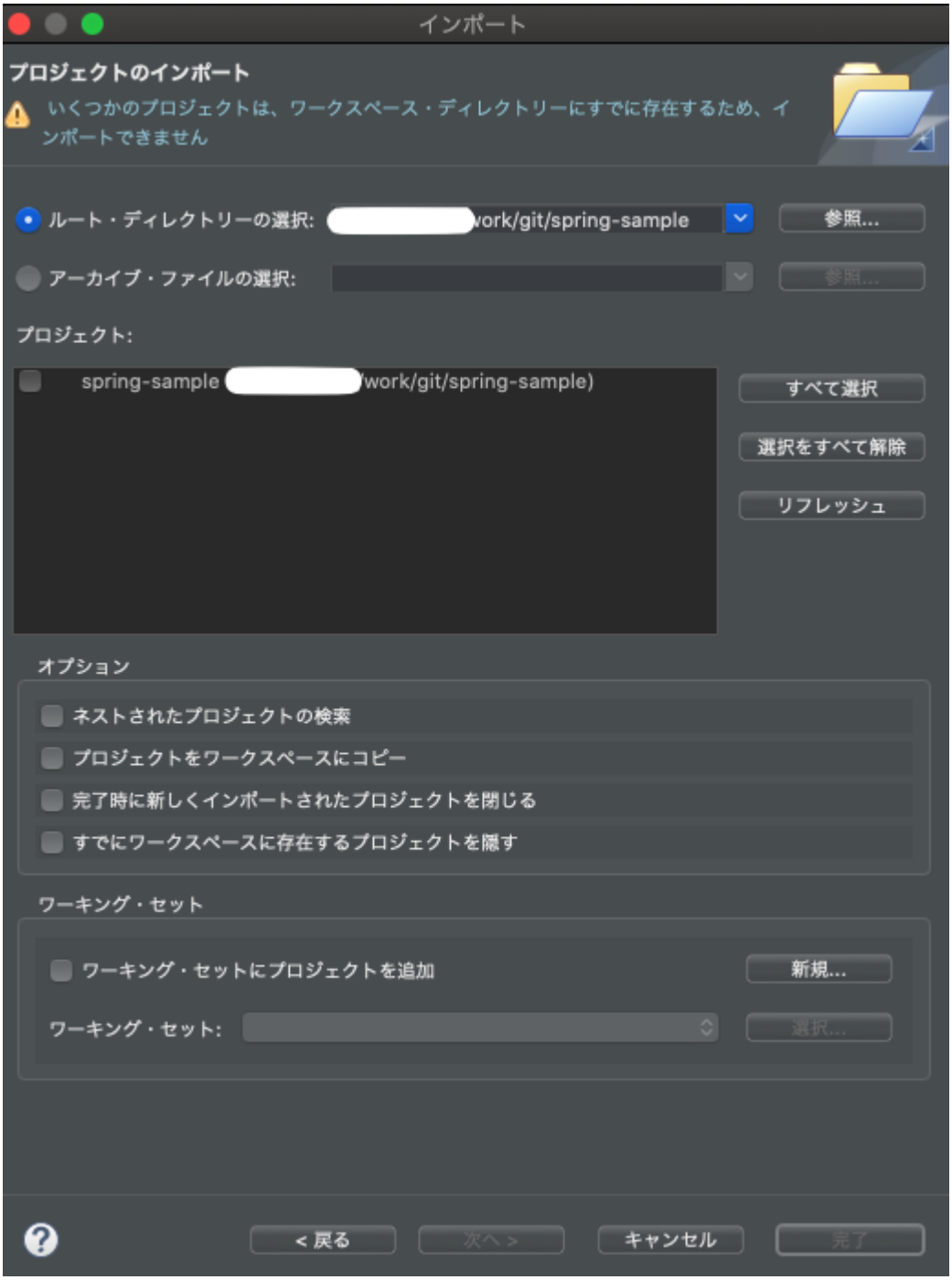
1. リポジトリーのクローン
   Gitのクライアントツール等でローカルにこのリポジトリをクローンする
   eclipseでインポートするためクローンする場所はメモしておく

2. eclipseにプロジェクトのインポート
   クローンしたリポジトリをeclipseでインポートする

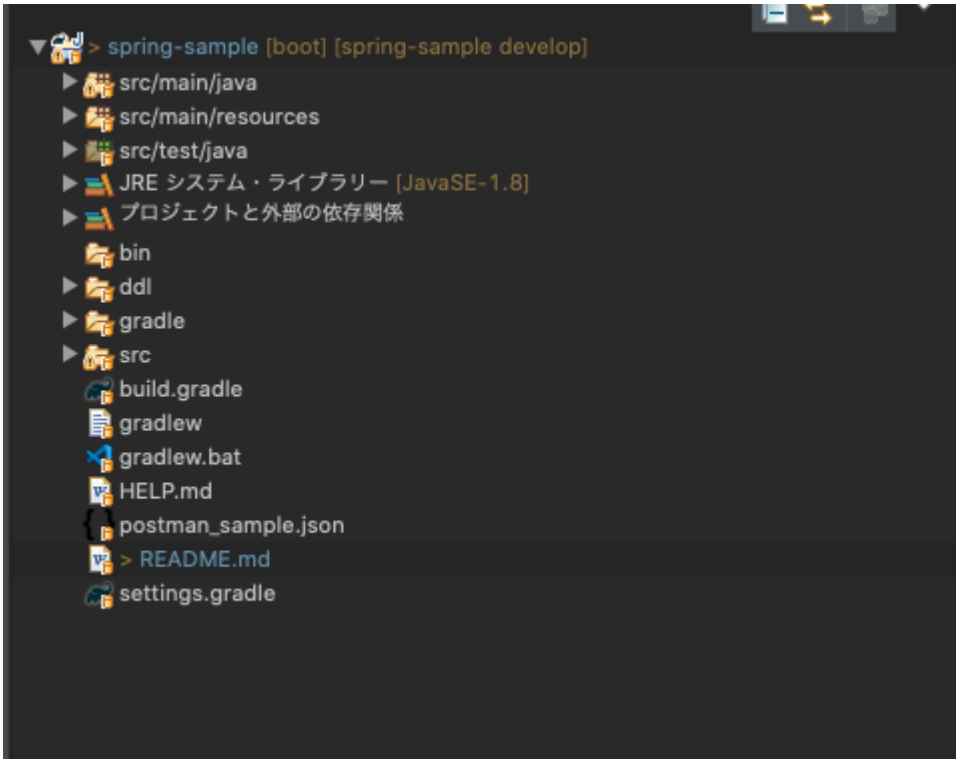
一般->既存プロジェクトをワークスペースへ



クローンしたプロジェクト(ディレクトリ)を選択する



インポート完了



主なリソースの構成

```
├── build.gradle  # gradleの依存関係をまとめるファイル
├── ddl    # テスト用のDB生成DDL
│   ├── 01.role.sql
│   ├── 02.database.sql
│   └── 03.tables.sql
├── gradle    # gradle関連ファイル(特に使用しない)
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradlew    # gradleバッチファイル、コマンドラインでビルド等する際に使用
├── gradlew.bat    # gradleバッチファイル、コマンドラインでビルド等する際に使用
├── postman_sample.json    # 試験用ツールのテンプレートPostman
├── settings.gradle    # gradle設定ファイル
└── src #プログラムファイル
    ├── main
    │   ├── java
    │   │   └── com
    │   │       └── example
    │   │           └── demo
    │   │               ├── SpringSampleApplication.java
    │   │               ├── api
    │   │               │   ├── organizations
    │   │               │   └── users
    │   │               │       ├── UserRestController.java
    │   │               │       └── UsersBean.java
    │   │               ├── config
    │   │               │   ├── JerseyConfig.java
    │   │               │   └── WebConfig.java
```

```
        |   |               └── domain
        |   |                   ├── entity
        |   |                   |   ├── organizations
        |   |                   |   |   ├── Organizations.java
        |   |                   |   |   ├── OrganizationsExample.java
        |   |                   |   |   └── OrganizationsKey.java
        |   |                   |   └── users
        |   |                   |       ├── Users.java
        |   |                   |       ├── UsersExample.java
        |   |                   |       └── UsersKey.java
        |   |                   ├── mapper
        |   |                   |   ├── organizations
        |   |                   |   |   └── OrganizationsMapper.java
        |   |                   |   └── users
        |   |                   |       └── UsersMapper.java
        |   |                   └── service
        |   |                       ├── organizations
        |   |                       └── users
        |   |                           ├── UsersService.java
        |   |                           └── UsersServiceImpl.java
        |   └── resources
        |       ├── application.properties
        |       ├── com    # ORマッパーのSQLファイル
        |       |   └── example
        |       |       └── demo
        |       |           └── domain
        |       |               └── mapper
        |       |                   ├── organizations
        |       |                   |   └── OrganizationsMapper.xml
        |       |                   └── users
        |       |                       └── UsersMapper.xml
        |       ├── generationConfig.xml    # MyBatisのORマッパー自動生成用の設定フ
    ァイル
        |       ├── generationConfig.xml.org
        |       ├── static
        |       └── templates
        └── test
            └── java
                └── com
```

ORマッパーの自動生成のやり方はこちらを参照

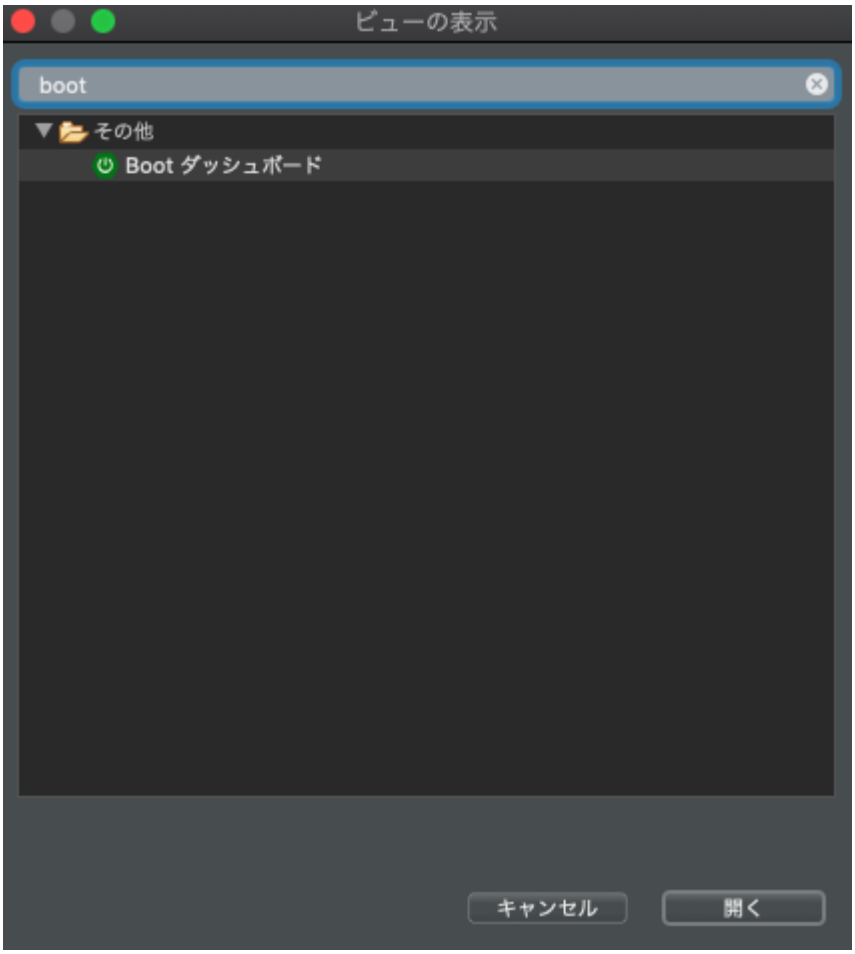
3. テスト用データベース作成
   クローンしたリポジトリ内にあるDDLを以下の順番で実行してデータベースを作成する

- 01.role.sql
- 02.database.sql
- 03.tables.sql

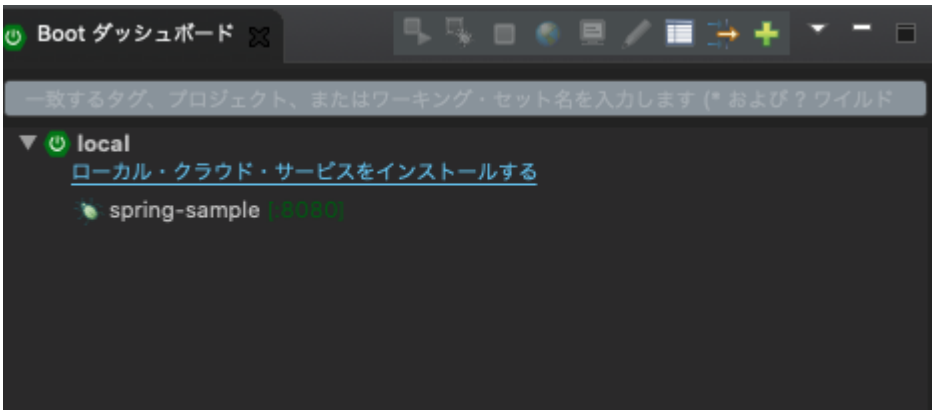
ユーザパスワードはspring/spring

4. アプリケーションの起動
   eclipseを用いたアプリケーションの起動

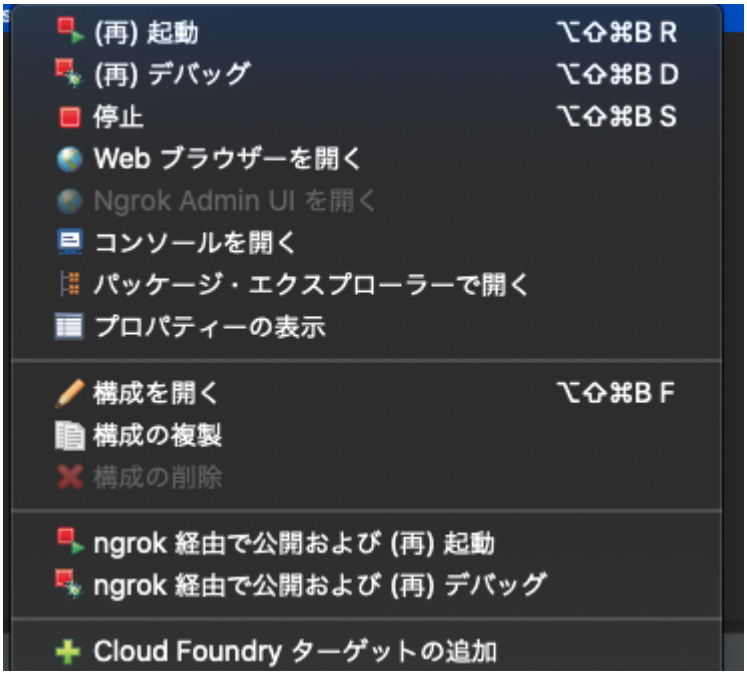
ビューの表示でBootダッシュボードを表示する



Bootダッシュボードが開くと対象のプロジェクトがリストアップされる

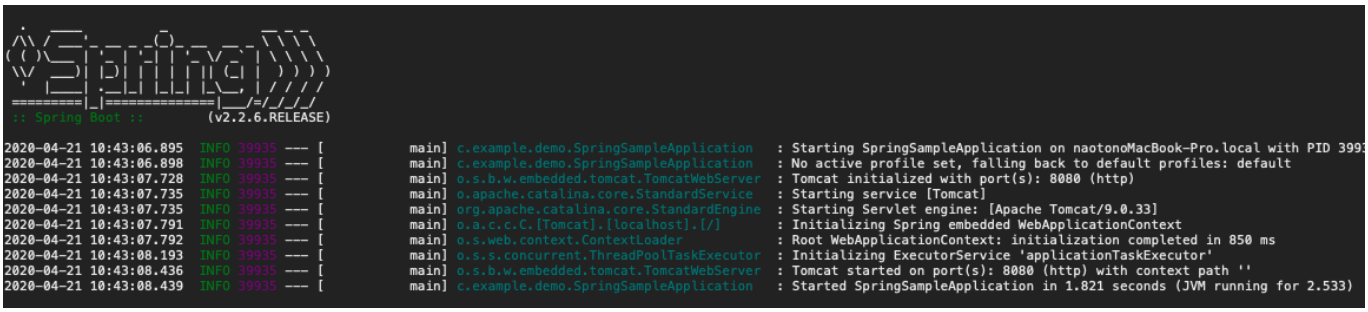


右クリックでデバッグ、もしくは通常起動をクリックするとアプリが起動する



以下のコンソールが出力されると起動完了

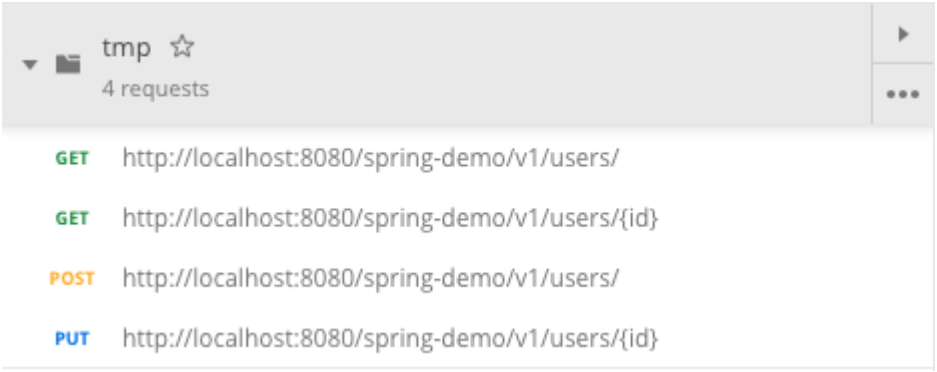
```
Started SpringSampleApplication in 1.821 seconds
```



5. 動作確認 アプリが起動したらRESTクライアントツールを用いて動作確認を実施する
   テストはChromeAppのPostmanを使用する(PostmanでなくてもCurl等のツールがあればそちらで問題無し)
   インストールはこちらを参照
   Postmanを使用する場合はリポジトリにある以下の定義がインポートして使用可能です。

```
postman_sample.json
```

インポートすると以下の定義が読み込まれます。



それぞれユーザの一覧取得、ID指定の取得、登録、更新です。

最後に

今回テーブルUsersとOrganizations２つ用意し、サンプルのAPIとしてUsersがあります。
基本的にはUsersを参考にすればOrganizationsのAPIも作成できると思います。
手順ではeclipse上で動作確認していますが、gradleでjarを生成すればTomcatが同梱されたモジュールが生成できるので、 スタンドアロンで動作させる事も可能です。モジュールの生成方法等はeclipseでやる方法やバッチファイルから実施する方法が あるので、やりやすい方を調べて試してください。