

JAVA開発のお作法

～勉強会のグラウンドルールとソフトウェア品質～

勉強会で作るアプリケーションにも品質指標があります。 まだあまり明確になっていませんが。。

品質特性

ソフトウェアの品質は**品質特性**によって評価されます。

- ・機能性 ... 従業員スキル管理
- ・信頼性 ... Junit、SpotBugsの利用
- ・使用性 ... JavaDocの記述
- ・効率性 ... Commitの活用
- ・保守性 ... インデントの統一
- ・移植性 ... Springフレームワーク



勉強会

「**グラウンドルール**」

資料:ソフトウェア開発へのSWEBOKの摘要 松本吉弘 オーム社 2005年、JIS X0129 【ISO/IEC9126】

今日のゴール(目指すところ)

勉強会のJAVAお作法を学んで、JAVA開発で実践できるようになる。



信頼性

(JUnit、SpotBugs)



共通処理(ユーティリティ)を実装したら、テストも実装してください。

信頼性向上の施策

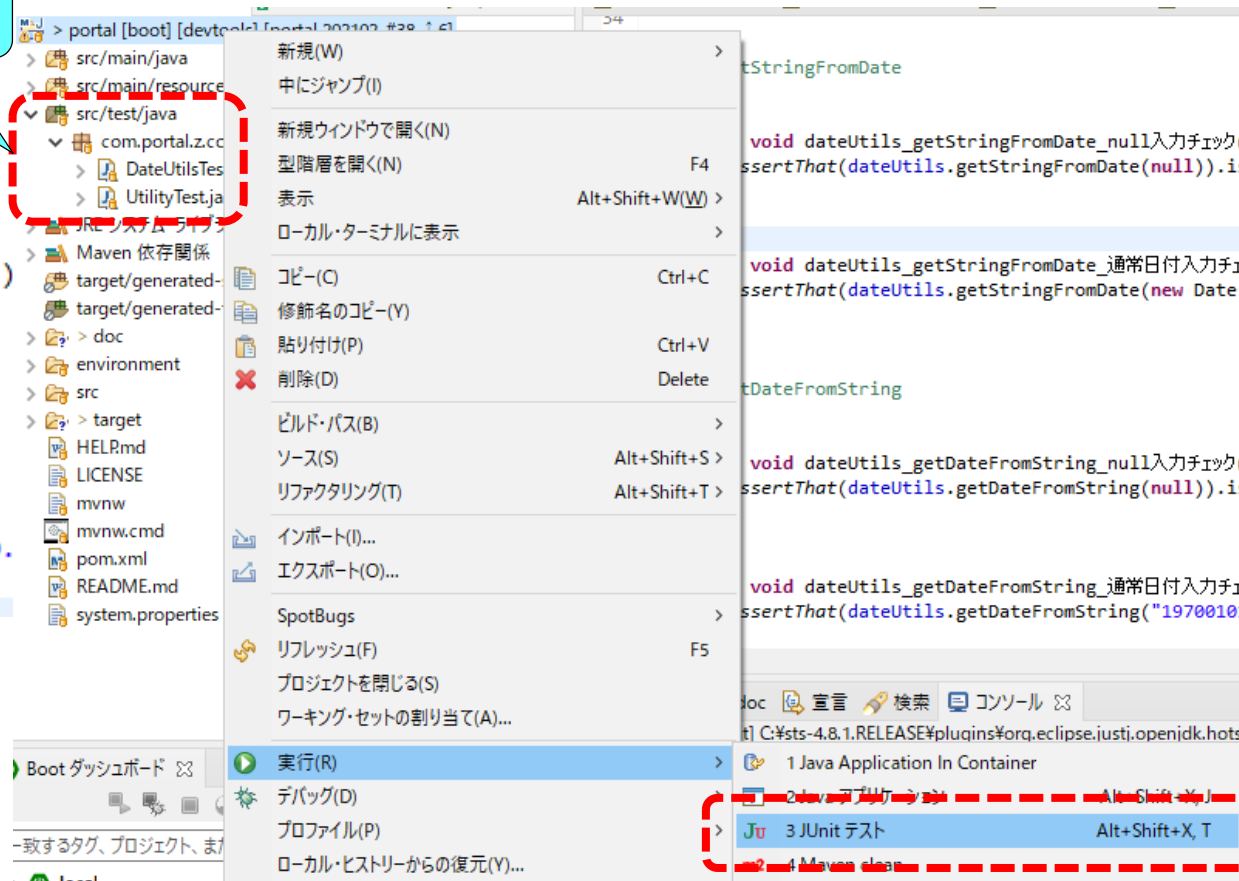
テスト実行

```
@Test
final void dateUtils_setStartDate_null入力チェック() {
    // 結果が1年ずれてしまう...
    assertThat(dateUtils.setStartDate(null)).isEqualTo("0001-01-02T23:41:01.000")
}

//
// setEndDate
//
@Test
final void dateUtils_setEndDate_通常日付入力チェック() {
    assertThat(dateUtils.setEndDate(new Date(0))).isEqualTo("1970-01-01T09:00:00.000")
}
```



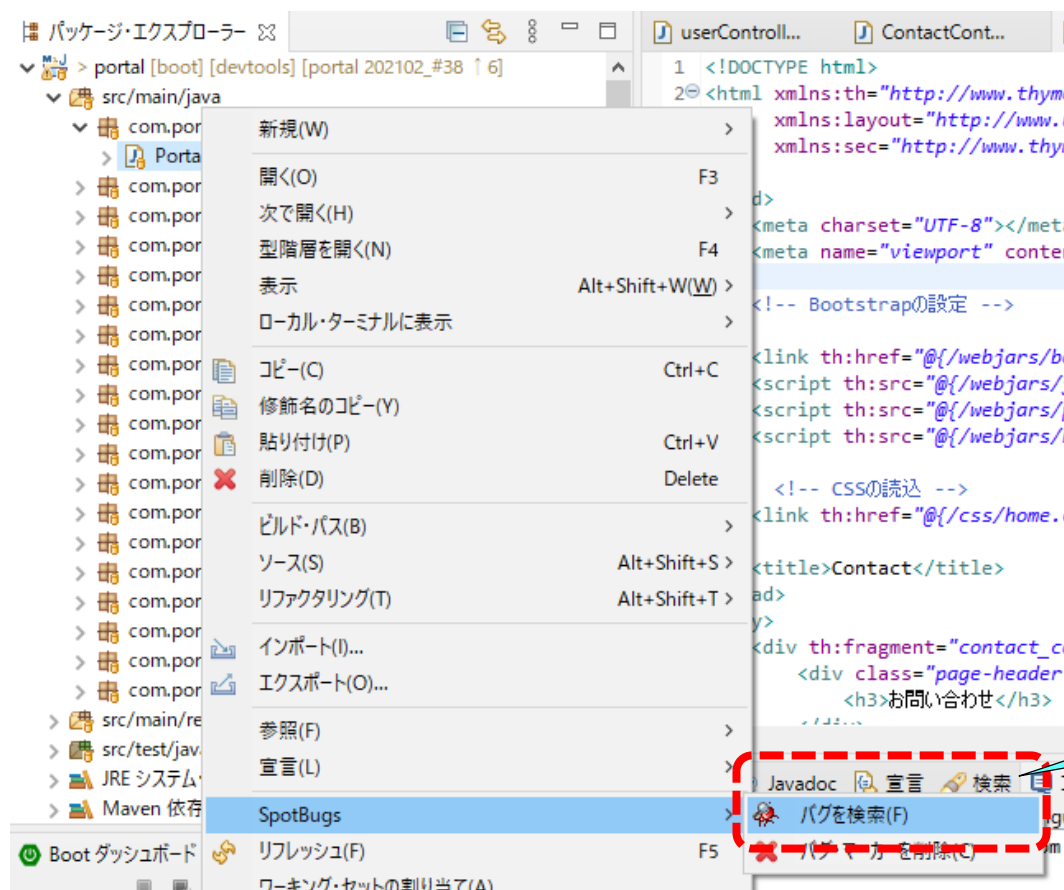
共通処理の不具合は全体への影響が大きいのでテストの記述を必須とします。



資料: 勉強会のWIKI

アプリケーションの信頼性を高めるために、バグの自動検出ツールを使います。(SpotBugs)

信頼性向上の施策



インストール方法は
「環境構築手順」に記
載されています。



資料:勉強会のWIKI

使用性

(JavaDoc)



各クラスとメソッドにJavaDocを記入して、仕様書の代わりとします。

使用性向上の施策

```
/**
 * ユーザー詳細画面のユーザー更新用処理.<BR>
 *
 * 画面から入力したユーザ情報でデータを更新する。
 *
 * @param form      入力用form
 * @param bindingResult 更新する情報
 * @param model      モデル
 * @return getUserList(model)
 */
@PostMapping(value = "/userDetail", params = "update")
// ユーザ詳細画面は更新も削除も/userDetailにPOSTするため、どちらが押されたかを判断するために、
// name属性の値をパラメータとして使っています
public String postUserDetailUpdate(@ModelAttribute @Validated(UpdateOrder)
    BindingResult bindingResult, Model model) {

    // 入力チェックに引っかかったらユーザー詳細画面に戻る
    if (bindingResult.hasErrors()) {
        log.info("入力チェックに引っかかったため、ユーザー詳細画面に戻る");
        return "userDetail";
    }

    // GETリクエストの場合は、ユーザー詳細画面に戻る
    return "userDetail";
}
```

Alt+Shift+Jで自動作成されます。

概要 パッケージ クラス 使用 階層ソリ 非推奨

概要: ネスト | フィールド | コンストラクタ | メソッド

postUserDetailUpdate

```
@PostMapping(value="/userDetail",
    params="update")
public java.lang.String postUserDetailUpdate(@ModelAttribute @Validated(UpdateOrder)
    BindingResult bindingResult, Model model) {

    ユーザー詳細画面のユーザー更新用処理。
    画面から入力したユーザ情報でデータを更新

    パラメータ:
    form - 入力用form
    bindingResult - 更新する情報
    model - モデル

    戻り値:
    getUserList(model)
}
```

[ファイル]-[エクスポート]-[Java]-[JavaDoc]で出力されます。

【ルール】

- ・クラスとパブリックなメソッドには必ずつける。
- ・@paramと@returnは必ず設定する
- ・最初にそのクラス(メソッド)の概要をかく
- ・注意事項があれば2行目以降に書く(nullの扱い、間違えそうなtrue/falseの判定など)

効率性

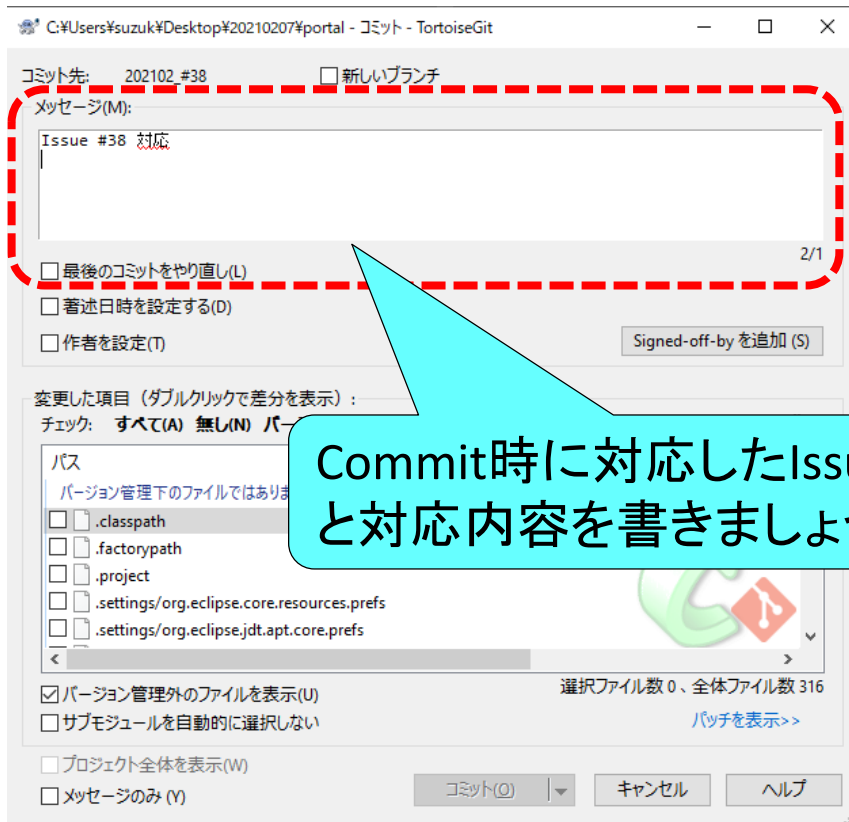
(Commitの活用)



Commitを工夫することで、ソースレビューや後からソースを追う作業が楽になります。

効率性向上の施策

実装が全部終わって最後にCommitするのではなく、区切り毎に随時Commitをしましょう。
⇒レビューの効率が上がります。




Commit時に対応したIssue番号と対応内容を書きましょう



【参考】
git commit するまえに考えるべき10のこと

保守性

(インデントの統一)



実装したソースをCommitする前に書式をそろえましょう。

保守性向上の施策

```
@PostMapping(value = "/userDetail", params = "update")
// ユーザ詳細画面は更新も削除も/userDetailにPOSTするため、どちらが押され
// name属性の値をパラメータとして使っています
public String postUserDetailUpdate(@ModelAttribute @Valid
    BindingResult bindingResult, Model model) {

    // 入力チェックに引っかかった場合、ユーザー詳細画面に戻る
    if (bindingResult.hasErrors()) {

        Log.info("入力エラー");

        // GETリクエスト用のメソッドを呼び出して、ユーザー詳細画面に戻ります
        return getUserDetail(form, model, "");
    }

    // Userインスタンスを作成
    User user = new User();

    // フォームからuserに値を代入
    user.setName(form.getName());
    user.setEmail(form.getEmail());
    // パスワードを設定
    user.setPassword(form.getPassword());
}
```

インデントがそろっているとソースが見やすくなるので、後から別の人がソースを修正するときに理解しやすくなる。

[Ctrl]+[Shift]+Fで書式がそろう。
※タブがスペースに置き換わるなど。



今日のゴール(目指すところ)

勉強会のJAVAお作法を学んで、JAVA開発で実践できるようになる。



ご清聴ありがとうございます