プログラマー向け成果物セルフチェックリスト

第1.0版

2019年3月26日

プログラマー向け成果物セルフチェックリスト

No	大分類	小分類	チェック項目	チェック内容	Nablarch依存有無	補足説明	処理方式毎のチェック有無			確認者	確認日	コメント
							画面	REST	バッチ			
1	共通	全般	ファイルの改行コード	・ファイルの改行コードがPJ指定の改行コードに統一されていること。		■例 Java、SQL、JSP、Shell、configなどのファイルすべてを共通でチェックする。	●	●	●			
2	共通	全般	ファイルの文字コード	・ファイルの文字コードがPJ指定の文字コードに統一されていること。		■例 Java、SQL、JSP、Shell、configなどのファイルすべてを共通でチェックする。	●	●	●			
3	Java	全般	"" or Nullの判定	・"" or Nullの判定には、Nablarchが提供する以下のAPIを使うこと。 ・値が設定されているかの確認 ⇒StringUtil#hasValue ・値が未設定かの確認 ⇒StringUtil#isNullOrEmpty ・無用な否定演算子を使わないこと。 NG例：if(!StringUtil.isNullOrEmpty(hoge)) この場合はStringUtil#hasValueを使うこと。	●	■値が設定されていることを確認する場合 誤： if (!"".equeal(value)) {} if (value != null) {} if (!StringUtil.isNullOrEmpty(value)) {} 正： if (StringUtil.hasValue()) {} ■値が未設定であることを確認する場合 誤： if ("".equeal(value)) {} if (value == null) {} if (!StringUtil.hasValue(value)) {} 正： if (StringUtil.isNullOrEmpty(value)) {}	●	●	●			
4	Java	全般	文字列結合	・文字列の結合でnullが含まれる場合に、"null"という文字列で結合しないこと。		■対応方法 nullとなる可能性のある文字列を結合する場合は、(Apache Commons) StringUtils#join 等のメソッドを利用し結合する。	●	●	●			
5	Java	全般	ファイル読み書き時のクローズ処理	・ファイル読み書き時は、以下を参照し、クローズ処理の有無を確認すること。 ①DataRecordResponseを使用して、HttpResponseに書き込みを実施した場合 →フレームワーク側でclose処理を実施するため、不要。 ②FileRecordWriterHolderを使用してファイル書き込みを実施した場合 →フレームワーク側でclose処理を実施するため、不要。 →ただし、FileRecordWriterHolderはバッチ処理の中でのみ使用可。（画面処理では使用不可。） ③DataRecordFormatterを使用してファイルの読み書きを実施した場合 →必ずclose処理を実施すること。 ④上記以外のクラス（Java標準のクラス）を使用してファイル読み書きを実施した場合 →必ずclose処理を実施すること。	●		●	●	●			
6	Java	全般	メソッド引数	・使用しない引数を定義しないこと。 ・メソッドに必要な最小の引数を定義すること。		■例 Dto内の1プロパティのみ使用する場合は、 Dtoを引数とせず、利用するプロパティのみを引数とする。	●	●	●			

7	Java	全般	booleanのメソッド引数	・メソッド引数にbooleanを渡し、その値で処理を分岐しないこと。		■理由 多くの場合は、呼び元で分岐し、メソッドを呼び分けた方が良いため。	●	●	●			
8	Java	全般	マジックナンバー	・マジックナンバーは定数化して適切な変数名を付与すること。			●	●	●			
9	Java	全般	抽象的過ぎる変数名	・抽象的過ぎる変数名を付けないこと。		■抽象的過ぎる変数名 NG例：Pattern PATTERN この変数名では、パターンの用途が不明確なため、意図を汲み取ることが出来る変数名にすること。 修正例：EXTRACT_HOSTNAME_PATTERN	●	●	●			
10	Java	全般	型を意識した変数名、メソッド名	・変数名やメソッド名に型を意識した名前を付けないこと。		■例 NG例：XXXMap、XXXStrなど。	●	●	●			
11	Java	全般	booleanを返すメソッドの名前	・isXXX や hasXXX , canXXX 等の名称にすること。 ・XXXに当たる部分は戻り値（ true/false ）が明確になるような名前を付けること。		■例 OK例：hasError →エラー時に true が返ることが想像できる。	●	●	●			
12	Java	全般	定数値との文字列比較	・hoge.equals(定数)ではなく、定数.equals(hoge)とすること。		■理由 hoge.equals(定数)ではNullPointerExceptionが発生する可能性があるため。	●	●	●			
13	Java	全般	0件であることの判定方法	・コレクションを扱う場合、Collection#isEmpty() を使用して判定を行うこと。		■理由 サイズを取得してから、0 であることを判定すると記述が冗長になるため。	●	●	●			
14	Java	全般	日付	・使用すべき日付の種類が誤っていないこと。		■例 業務日付、システム日付など。	●	●	●			
15	Java	全般	メソッド呼び出しの入れ子	・メソッド呼び出しの入れ子は最大２回までにすること。		■理由 メソッド引数にメソッド呼び出しを含めすぎると可読性が下がるため。 NG例： setZZZ(convertXXX(getXXX(hoge,huga),getYYY(moge,moga).toString().subString(0,12))); OK例： String foo = getXXX(hoge,huga); String bar = getYYY(moge,moga).toString().subString(0,12); setZZZ(convertXXX(foo, bar));	●	●	●			
16	Java	全般	メソッドの並び順	・public protected privateの順番に記載すること。			●	●	●			
17	Java	Action	エラー時の遷移先	・排他エラーやバリデーションエラー時のフォワード先が正しく設定されていること。			●					
18	Java	Action	自動設定項目	・@CurrentDateTimeなど、自動で値が設定されるものは、個別の業務ロジックにて設定しないこと。	●		●	●	●			

19	Java	Action	インスタンス変数	・Actionにインスタンス変数を持たせないこと。	●	■例外 バッチの出力件数用カウンタ等、インスタンス変数で実現する必要がある場合は除く。	●	●	●			
20	Java	Form	Formのドメイン定義	・設計書上に指定されているドメイン定義を確認し、ドメイン定義書で定義された精査処理が正しく実装されていること。			●	●	●			
21	Java	DBアクセス	排他制御	・更新処理時にUniversalDaoを使用した排他制御を実装すること。	●	■実装方法 1.更新対象のレコードをEntityに格納する。 ※画面の場合、初期表示時に上記のEntityはセッションストアに格納し、 リクエスト間を持ち回る。 2.更新する項目だけEntityを上書きする。 ※バージョン番号はフレームワークが自動で更新するため、Entityの上書きは不要。 3.UniversalDao.update(entity)で更新処理を実行する。	●	●				
22	Java	DBアクセス	DBアクセス	・DBアクセスは原則UniversalDaoを使用すること。 ・同じSQL文を複数回発行する場合は、バッチ実行メソッドを使用すること。（UniversalDao#batchInsertなど）	●	■UniversalDaoで提供している操作 ・登録/一括登録 ・主キーを指定した更新/一括更新 ※排他制御を行う更新時のみ使用すること ・主キーを指定した削除/一括削除 ・主キーを指定した検索 ・SQL文による検索	●	●	●			
23	Java	DBアクセス	複数件の検索結果	・複数件の結果を取得できるSQLを発行した場合は、 件数を正しく判定すること。 ・1件でよい場合は、SQLで1件のみ取得するように 実装すること。 ・複数件結果が返却されるSELECTに対して、 SqlResultSet#get(0)を実行しないこと。	●	■理由 複数件結果が返却されるSELECTに対してSqlResultSet#get(0)を実行している場合、検索条件指定が漏れている、設計時の考慮が足りていない等の不具合の可能性があるため、仕様を確認する。	●	●	●			
24	Java	DBアクセス	検索結果0件	・検索結果が0件となる場合の実装が考慮されていること。 仕様を確認し、下記いずれかであることを確認する。 ①仕様上、0件があり得る場合 ・設計書に記載されている0件時のハンドリングが実装されていること。 例）画面上でユーザー通知を行う、エラーハンドリングを行う。 ②仕様上、0件があり得ない場合 ・0件時のハンドリングが実装されていないこと。	●	■検索結果0件のハンドリング例 ・検索結果がありませんと画面に表示する。 ・アプリケーション例外を投げる。	●	●	●			
25	Java	DBアクセス	一覧検索の検索結果件数の取得方法	・UniversalDao#countBySqlFileを使用すること。 ・SqlResultSet#isEmptyでは現在ページの結果件数判定となるため、このメソッドを使用しないこと。	●		●	●	●			
26	Java	DBアクセス	SqlRowからの値取得	・SqlRow#get()してキャストするのではなく、適切な取得メソッドを使用すること。	●	■例 SqlRow#getString()やSqlRow#getBigDecimal()など。	●	●	●			
27	JSP	全般	ボタンサブミットの制御	・以下のカスタムタグを利用する場合は、type="button"を指定して、エンターキーの押下でサブミットが実行されないようにすること。 ・n:button ・n:popupButton ・n:downloadButton	●	■例 type="button"をタグの属性に指定する。 NG例： <n:button uri="hoge/000">fuga</n:button> OK例： <n:button type="button" uri="hoge/000">fuga</n:button>	●					

28	JSP	全般	入力項目の子画面連携	・画面で入力された内容をリクエストパラメータ名を変更して子画面に送信する場合、n:changeParamNameを使用すること。	●	■理由 n:paramによりリクエストパラメータ名を変更して送信すると、送信される値が 画面の初期表示時点の値となるため。	●					
29	JSP	全般	アプリケーション外へのサブミット	・アプリケーション外へのサブミットには、n:submitではなく、n:aを使用すること。	●	■理由 アプリケーション外へのサブミットの場合はリクエストIDが存在せず、submitリンクにすると全て非活性になってしまうため、n:aタグを使用する必要がある。 ■例 ・子画面から親画面に値を連携する時 ・閉じるボタン ・ページ内リンク	●					
30	JSP	全般	画面に入力項目が一つだけ場合	・入力項目について、<n:text>のような<input type="text">形式の項目を一つだけ実装している画面にて、テキストボックスにフォーカスをあてENTERを押下するとエラーになるため、display:none & disable のダミー用テキストボックスを表示しているテキストボックスより上部に配置すること。	●	■例 ・検索条件(テキストボックス)と検索ボタンしか存在しない画面 ・検索条件１（ラジオボタン）、検索条件２(テキストボックス)および検索ボタンしか存在しない画面 ※この現象はブラウザ起因で発生するため、Nablarchのバージョンには依存しない。	●					
31	テスト	全般	期待値のアサート	・アサート対象に応じて、適切なアサートメソッドを呼び出すこと。		■例 ・assertEquals系 ・assertSqlRowEquals ・assertSqlResultSetEquals	●	●	●			
32	テスト	全般	テーブルアサート	・テーブルアサートにはEXPECTED_COMPLETE_TABLEを使用してアサートすること。	●	■理由 EXPECTED_TABLEを使用した場合、省略されたカラムは比較対象外となるが、EXPECTED_COMPLETE_TABLEを使用した場合は、省略されたカラムにはデフォルト値が格納されているものとして比較が行われる。	●	●	●			
33	テスト	全般	例外のテスト	・try-catchで例外送出を確認するのではなく、@Test(expected=IllegalArgumentException.class)を使用すること。ただし、Exceptionの内容を確認したい場合は、ExpectedExceptionを使用すること。 ・１メソッドで複数種類の例外が発生する場合は、例外種別毎にテストメソッドを分けること。	●		●	●	●			
34	テスト	全般	レビュー対象物	・レビュー対象に漏れが無いことを確認すること。		■理由 以下のファイルは漏れやすいため要注意。 ・SQLファイル ・XMLファイル ・fmtファイル	●	●	●			

●：依存有