例外ハンドリング

ハンドリング方針早見表（APチーム提供）

表示隠す

No.		種類	発生原因	遷移先	発生後の動作	遷移先ajax	発生後の動作Ajax	ハンドリング	備考	発生例など
1	コントロール	前提条件チェック	本来あるはずのパラメータやSessionが無いなどのチェック	共通エラー	共通エラー画面に遷移し、固定のエラーメッセージを表示する	共通エラー	非Ajaxと同じ	BadRequestExceptionをthrow catchしない		予約導線だと、記録番号が存在しない場合など
2		画面遷移違反（不正操作防止）	複数Windowを立ち上げて、両方で操作した場合などを防ぐ	共通エラー	共通エラー画面に遷移し、固定のエラーメッセージを表示する	共通エラー	非Ajaxと同じ	UnRecoverableBusinessExceptionをthrow catchしない	URL直接入力においては照会系でURL直接入力された際に、ログイン画面に飛ばしてあげるので、これは別途。 複数タブ操作に対する制御はチケット管理なので参照してください。 http://10.224.4.130:8080/redmine/issues/3258	上記と合わせて考えるべきだが、想定外の画面からの遷移とかをチェックする？
3		単項目チェック		自画面	画面上部に入力チェックに対応するエラーメッセージを表示する。	自画面	・画面上部にメッセージを出す(参考) ・Ajax表示領域内にエラーを出す(参考)	例外はthrowしない。 入力チェック参照	(国内専用メモ)冨永さんスケジュールで決めるが、Ajaxで出すエラーの表出箇所等は機能によって違うらしいので確認しておいてほしい	未入力チェックとか Formクラスのアノテーションによるチェック
4		相関チェック	複数の入力パラメータの相関関係によるチェックエラー	自画面	画面上部に入力チェックに対応するエラーメッセージを表示す	自画面	〃	例外はthrowしない。 入力チェック参照		姓がカナで名が漢字 FormValidatorによるチェック

					る。					
5		Session切れ	有効期間超過などにより、Sessionが取れない	共通エラー	共通エラー画面に遷移し、固定のエラーメッセージを表示する	共通エラー	非Ajaxと同じ	FWで実施するので何もしない	基盤Tが提供 検知した場合は固定文言で一律共通エラー画面に遷移。 検知機能を入れたくない画面もあるため、アノテーション付与等で検知除外設定できるようにする。	
6	サービス	APIエラー	APIからエラーコードが返された場合で、先に進めるエラー	自画面	APIのエラーコードと返却されたパラメータをもとにエラーメッセージを生成し、自画面を再表示画面上部に生成したエラーメッセージを表示する	自画面	No.4と同じ	RecoverableBusinessExceptionをthrow ControllerでCatch	(国内専用メモ)冨永さんスケジュールで決めるが、Ajaxで出すエラーの表出箇所等は機能によって違うらしいので確認しておいてほしい	排他エラー（記録更新中） AIR到着時間と電車の時間が短い場合とか（カート更新時に発生するんので予約では発生しないかもですが） 出発後の解約など
7			APIからエラーコードが返された場合で、先に進めないエラー	共通エラー	APIのエラーコードと返却されたパラメータをもとにエラーメッセージを生成し、共通エラー画面に遷移し、生成されたエラーメッセージを表示する。	共通エラー	非Ajaxと同じ	UnRecoverableBusinessExceptionをthrow	(国内専用メモ)冨永さんスケジュールで決める	API側でのシステムエラーサービス利用不可の場合など
8		業務エラー	業務チェックで判定されるエラー（先に進めるエラー）	自画面	画面上部に入力チェックに対応するエラーメッセージを表示する。	自画面	No.4と同じ	RecoverableBusinessExceptionをthrow ControllerでCatch	(国内専用メモ)冨永さんスケジュールで決めるが、Ajaxで出すエラーの表	排他エラー（記録更新中）

									出箇所等は機能によって違うらしいので確認しておいてほしい	
9			業務チェックで判定されるエラー（先に進めないエラー）	共通エラー	共通エラー画面に遷移し、エラー内容に対応するエラーコードとメッセージをを表示する	共通エラー	非Ajaxと同じ	UnRecoverableBusinessExceptionをthrow	(国内専用メモ)冨永さんスケジュールで決めるが、Ajaxで出すエラーの表出箇所等は確認しておいてほしい	
10	Gateway	通信エラーとか	外部I/Fで通信の失敗など	共通エラー	Gatewayでの外部システムへの通信失敗	共通エラーor自画面	非Ajaxと同じ	Runtime系のExceptionなので自分ではthrowしない 基本的にcatchしない。 予約系の一部ではcatchして自画面遷移させる	FW一任。 固定文言一律だす 画面側で処理を続行したい場合（Runtime系もcatchしてハンドリングしたい場合）は、共通部品を使ってね	
11	その他	システムエラー	バグとか	共通エラー	予想外のExceptionの場合	共通エラー	非Ajaxと同じ	Runtime系のExceptionなので自分ではthrowしない 基本的にcatchしない	FW一任。 固定文言一律だす	
12		設定不正	プロパティから取り出した内容がおかしい	共通エラー	共通エラー画面に遷移し、固定のエラーメッセージを表示する	共通エラー	非Ajaxと同じ	何もしない	FW一任。 固定文言一律だす	取り出した内容がnull 取り出した内容が制御値として使えない
13	Ajax	通信エラーとか	Ajax通信において、ステータスコード200で返ってこなかった場合全部	-	-	-	ダイアログを出すor 最終的に静的ページにリダイレクトする (共通JSで)	基盤が検討中	Ajaxタイムアウト設定と同様であとから差し込める？ ダイアログを出す方針だが、静的ページだせるなら一	

									律静的ページにリダイレクトが一番楽。要確認。	

例外の分類

A, オペレータの再操作(入力値の変更など)によって、発生原因が解消できる例外

基本的にアプリケーションコードで例外をハンドリングし、例外処理を行う。
Ajaxリクエストの処理の場合はRecoverableBusinessExceptionについてはフレームワークに任せる。(アプリケーションコードでハンドリングしない)

該当する例外の種類

- ビジネス例外 (RecoverableBusinessException)
- 正常稼働時に発生するライブラリ例外 (e.g. DBの一意制約違反)

B, オペレータの再操作(入力値の変更など)によって、発生原因が解消できない例外

フレームワークで例外をハンドリングし、例外処理を行う。

該当する例外の種類

- ビジネス例外 (UnRecoverableBusinessException)
- システム例外
- 予期しないシステム例外 (e.g. DBサーバダウン、バグ)
- 致命的なエラー (e.g. OutOfMemoryError)

C, クライアントからの不正リクエストにより発生する例外

フレームワークで例外をハンドリングし、例外処理を行う。

該当する例外の種類

- リクエスト不正時に発生するフレームワーク例外
- Controllerにおける不正リクエスト検証により送出する例外

例外の種類

ビジネス例外

ビジネスルールの違反を検知したことを通知する例外 。
本例外は、サービス層で発生させる。
アプリケーションとして想定される状態なので、システム運用者による対処は不要。

例

- 旅行を予約する際に予約日が期限を過ぎている場合
- 商品を注文する際に在庫切れの場合

該当する例外クラス

- jp.co.anas.atc.fw.core.exception.RecoverableBusinessException
  - 入力値の変更などによって発生原因が解消できるビジネス例外 (通常のビジネス例外)
  - この例外は基本的にControllerでcatchして画面に表示する (Ajaxリクエストの処理ではcatchしない)
- jp.co.anas.atc.fw.core.exception.UnRecoverableBusinessException
  - 入力値の変更などによって発生原因が解消できないビジネス例外
  - 一連の業務フローをやり直す必要がある状態を検知した場合に送出する
  - この例外は基本的にControllerでcatchせず、フレームワークで例外をcatchし共通エラー画面に表示する

UnRecoverableBusinessException及び、そのサブクラスをControllerでcatchしなかった場合は、フレームワークで例外をcatchし共通エラー画面に表示する。
共通エラー画面に表示されるエラーメッセージは例外クラスが保持するメッセージコード、メッセージパラメータを基に構築したメッセージとなる。
なお、エラーメッセージの構築時にメッセージパラメータはHTMLエスケープされる。

システム例外

システムが、正常稼働している時に、発生してはいけない状態を検知したことを通知する例外 。
本例外は、主に共通部品(SharedService, Gateway, e.t.c.)で発生させる。
システム運用者による対処が必要となる。

例

- 事前に存在しているはずのマスタデータ、ディレクトリ、ファイルなどが存在しない場合
- フレームワーク、ライブラリ内で発生する検査例外のうち、システム異常に分類される例外を捕捉した場合(ファイル操作時のIOExceptionなど)
- 連携先APIがシステム例外相当のレスポンスを返してきた場合、接続できなかった場合

該当する例外クラス

- org.terasoluna.gfw.common.exception.SystemException
  - 遷移先のエラー画面や、HTTPレスポンスコードを細かく分ける場合は、SystemExceptionを継承した例外クラスを作成すること
  - 第2引数のメッセージはログに出力するメッセージであり、画面には表示しない
- jp.co.anas.atc.fw.core.exception.SimpleSystemException
  - エラーコードを指定しないSystemException

基本的にアプリケーションコードでcatchすることは禁止。
フレームワークで例外をcatchし共通エラー画面に表示する。
共通エラー画面に表示されるエラーメッセージは例外クラスごとに固定のメッセージとなる。

Controllerにおける不正リクエスト検証により送出する例外

通常の画面遷移、Ajaxリクエストではあり得ない、リクエスト内容の不正を検知したことを通知する例外。
URLを書き換えての直接リクエストや、リクエスト内容の改竄などをされない限り発生しないもの。
本例外は、プレゼンテーション層で発生させる。
原因は、クライアント側に存在するため、システム運用者による対処は不要である。

該当する例外クラス

- jp.co.anas.atc.fw.web.exception.BadRequestException

基本的にアプリケーションコードでcatchすることは禁止。
フレームワークで例外をcatchし共通エラー画面に表示する。
共通エラー画面に表示されるエラーメッセージは固定のメッセージとなる。

アプリケーション固有の例外クラスを作成する場合の留意事項

- 汎用的な例外クラスであれば、AP基盤に連絡すること
- アプリ固有の例外クラスであれば、所定のパッケージに作って構わない
  - 但し、必ずBusinessExceptionかSystemExceptionを継承すること

実装方法

RecoverableBusinessExceptionを送出する方法

エラーメッセージが一つの場合

次のコンストラクタを使用する。

- RecoverableBusinessException(String, Object...)
- RecoverableBusinessException(Throwable, String, Object...)}

コード例

```
throw new RecoverableBusinessExceptior"エラーメッセージID", "メッセージパラメータ1", 777, "メッセージパラメータ3");
```

エラーメッセージが複数の場合

次のコンストラクタを使用する。

- RecoverableBusinessException(ResultMessages)}
- RecoverableBusinessException(ResultMessages, Throwable)}

コード例

```
ResultMessages messages = ResultMessages.error();
messages.add(ResultMessage.fromCode("エラーメッセージID", "メッセージパラメータ1", 777, "メッセージパラメータ3"));
messages.add(ResultMessage.fromCode("もう一つのエラーメッセージID"));
throw new RecoverableBusinessException(messages);
```

留意事項

- ResultMessages.error()**以外のファクトリメソッドは使用しないこと** (e.g. ResultMessages.info())
- ResultMessage.fromText(String)**は使用しないこと**

要求送信元の画面の特定箇所にビジネス例外のメッセージを表示する方法 (画面表示リクエストの場合)

ControllerにてServiceが送出するBusinessExceptionをcatchして、modelに詰める。

```
try {
        final Foo foo = service.findFoo(fooId);
                    model.ad"foo", foo);
} catch final RecoverableBusinessException e) {
                    model.addAttribute(e.getResultMessages());
}
```

JSP上でmessagesPanelタグを配置することで、modelに詰めたエラーメッセージを表示する。

```
<atp-ui:messagesPanel />
```

※ messagesPanel タグは atd-csm-web, ati-csm-web に配置されているタグファイル

なお、表示する箇所が複数の場合(処理によって切り替える必要がある場合)は、次の様にmodelAttribute名を指定することで対応可能。

```
try {
        final Foo foo = service.findFoo(fooId);
                    model.ad"foo", foo);
} catch final RecoverableBusinessException e) {
                    model.ad"fooResultMessages", e.getResultMessages());
}
```

```
<atp-ui:messagesPanel messagesAttribute="${fooResultMessages}" />
```

要求送信元の画面の特定箇所にビジネス例外のメッセージを表示する方法 (Ajaxリクエストの場合)

Ajaxエラーハンドリングを参照のこと

ビジネス例外起因で特定の画面に遷移させる方法

例えば、アカウントロック状態になった時に共通エラー画面ではなく特定の画面に遷移させたい場合は、
BusinessExceptionを継承したAccoutLockExceptionという様な独自の例外クラスを作成し、サービス層で送出するようにする。

Controller側では次の様に、その例外をcatchして特定の画面に遷移させる。

```
try {
                    foo = service.login(userId, password);
} catch final AccoutLockException e) {
        return "redirect:アカウントロック画面のURLパス";
} catch final RecoverableBusinessException e) {
                    model.addAttribute(e.getResultMessages());
}
```

参考

- 4.2. 例外ハンドリング — TERASOLUNA Server Framework for Java (5.x) Development Guideline 5.3.0.RELEASE documentation
    - 方針や例外クラスはTERASOLUNAをベースにしている