関係社外秘

東京ITスクール
TOKYO IT SCHOOL

チーム開発演習

シェアードショップ

通販システム

詳細設計書

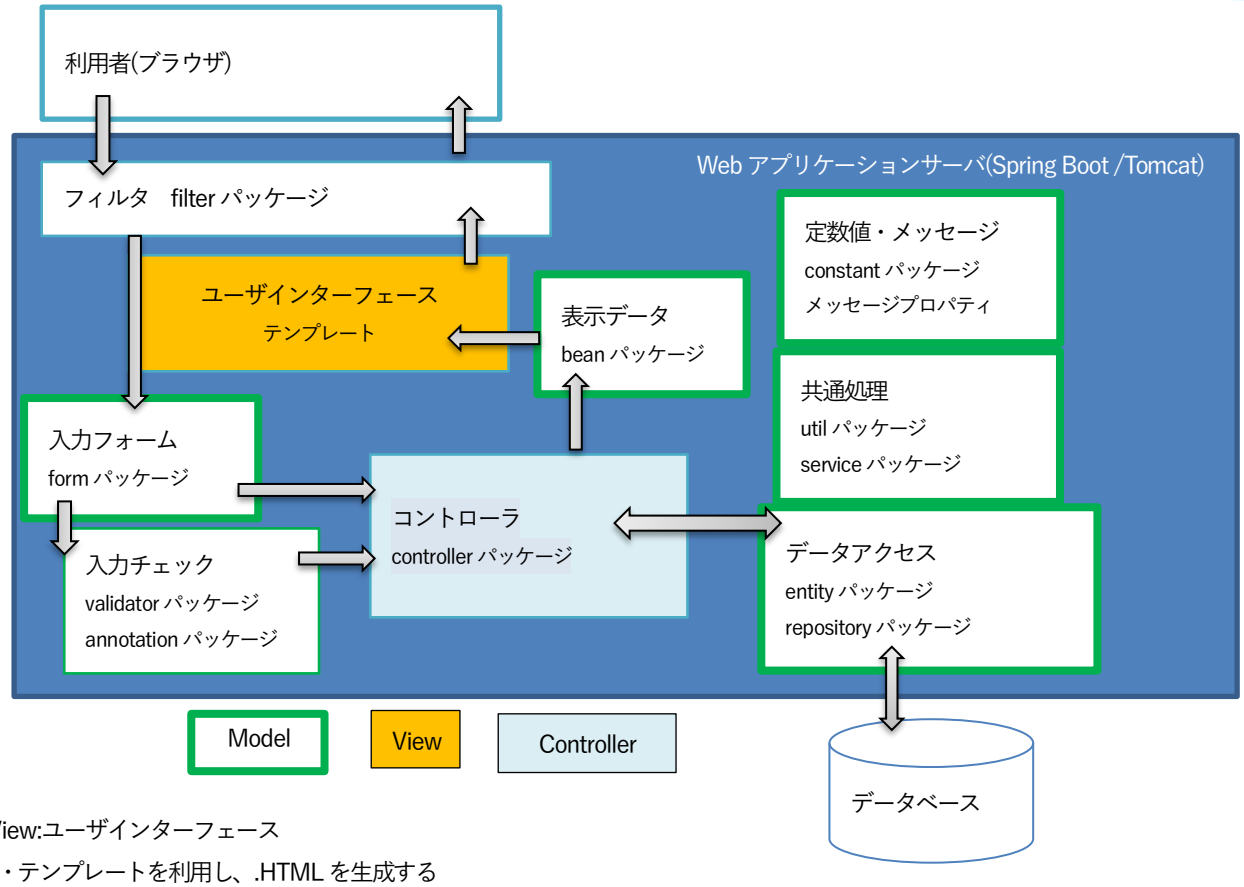
目次

1 はじめに

本書では、架空の顧客からの要望を記載しています。
初学者向け研修で利用することを考慮して、一般的な詳細設計書の内容よりも量と粒度を抑えてあります。ご留意ください。

2 プログラム構成

2 - 1 プログラム全体構成



(1)View:ユーザインターフェース

・テンプレートを利用し、.HTML を生成する
・画面間の情報の受渡は、スコープを利用する
・form タグの hidden 属性の利用は最小限に抑える

(2)Controller:コントローラ

・一般会員用と管理者(運用管理者・システム管理者)それぞれに各機能用コントローラを用意する
・@RequestMapping の記述により、利用者からのリクエストに対応するメソッドを用意する
・他のクラスと連携し、各機能を実装する

(3)Model:入力フォーム

・入力画面に合わせて、Form クラスを用意する
・Form クラス内で入力値チェックのアノテーションを指定する

(4)Model:入力チェック

・データベースにて一意制約のある入力項目について、重複チェッククラスを用意する
・Form クラスにて該当項目に独自アノテーションを指定する

⑸Model:表示データ

・表示データや DB から取得した情報保持をするために Bean クラスを用意する

⑹Model:共通処理、定数値・メッセージ

・共通で利用する処理は、@Service を付与したクラスで用意する
・フィルタで共通で利用する処理は、static メソッドで定義したクラスで用意する
・共用する定数値等を static フィールドで定義する
・表示メッセージは、messages.properties ファイルに定義する

⑺Model:データアクセス

・データベース操作のためのリポジトリクラスを用意する
・データベースとのデータ授受のためのエンティティクラスを用意する

⑻フィルタ

・アクセス制御、ログイン状態チェックなどを行うフィルタを用意する

2 - 2 パッケージ一覧

Spring Framework を利用し、下記のパッケージ構成とする。

パッケージ名(アルファベット順)	利用用途
jp.co.sss.shop. annotation	独自アノテーション
jp.co.sss.shop.bean	JavaBeans 表示用データ格納用
jp.co.sss.shop.config	Spring Boot の設定
jp.co.sss.shop.controller.admin.category	管理者用カテゴリ情報管理機能用コントローラ
jp.co.sss.shop.controller.admin.item	管理者用商品情報管理機能用コントローラ
jp.co.sss.shop.controller.admin.order	管理者用注文情報管理機能用コントローラ
jp.co.sss.shop.controller.admin.user	管理者用会員情報管理機能用コントローラ
jp.co.sss.shop.controller.client.basket	一般会員用買い物かご機能用コントローラ
jp.co.sss.shop.controller.client.item	一般会員用商品情報管理機能用コントローラ
jp.co.sss.shop.controller.client.order	一般会員用注文登録、表示機能用コントローラ
jp.co.sss.shop.controller.client.user	一般会員用会員機能用コントローラ
jp.co.sss.shop.controller.login	ログイン、ログアウト機能用コントローラ
jp.co.sss.shop.entity	データベース処理用エンティティ
jp.co.sss.shop.filter	フィルタ処理
jp.co.sss.shop.form	入力フォームの値授受用
jp.co.sss.shop.repository	データベース処理用インターフェース
jp.co.sss.shop.serivce	ビジネスロジック、Bean 生成・コピー
jp.co.sss.shop.util	固定値定義、共通処理
jp.co.sss.shop.validator	入力値のチェック処理

2-3コントローラの構成

・網掛け部分は、現行システムで実現済み

画面 ID	画面名	URL パターン	ファイル名
1-1-1-1	ログイン画面	/login	LoginController
-	ログアウト	/logout	LogoutController
1-3-1-3 1-3-1-4	管理者用メニュー画面	/admin/menu	LoginController
2-1-1-1 2-1-1-2	トップ画面	/	ClientItemShowController
2-1-2-1 2-1-2-2	商品一覧画面 カテゴリ別商品一覧画面	/client/item/list/{sortType} /client/item/list/{sortType}?categoryId={id}	ClientItemShowController
2-1-1-3	商品一覧画面	/admin/item/list	AdminItemShowController
2-2-1-1 2-2-1-2	商品詳細画面	/client/item/detail/{id}	ClientItemShowController
2-2-1-3	商品詳細画面	/admin/item/detail /{id}	AdminItemShowController
2-3-1-3	商品登録入力画面	/admin/item/regist/input	AdminItemRegistController
2-3-2-3	商品登録確認画面	/admin/item/regist/check	AdminItemRegistController
2-3-3-3	商品登録完了画面	/admin/item/regist/complete	AdminItemRegistController
2-4-1-3	商品変更入力画面	/admin/item/update/input	AdminItemUpdateController
2-4-2-3	商品変更確認画面	/admin/item/update/check	AdminItemUpdateController
2-4-3-3	商品変更完了画面	/admin/item/update/complete	AdminItemUpdateController
2-5-1-3	商品削除確認画面	/admin/item/delete/check	AdminItemDeleteController
2-5-2-3	商品削除確認画面	/admin/item/delete/complete	AdminItemDeleteController
3-1-1-3	カテゴリー覧画面	/admin/category/list	AdminCategoryShowController
3-2-1-3	カテゴリ詳細画面	/admin/category/detail/{id}	AdminCategoryShowController
3-3-1-3	カテゴリ登録入力画面	/admin/category/regist/input	AdminCategoryRegistController
3-3-2-3	カテゴリ登録確認画面	/admin/category/regist/check	AdminCategoryRegistController
3-3-3-3	カテゴリ登録完了画面	/admin/category/regist/complete	AdminCategoryRegistController
3-4-1-3	カテゴリ変更入力画面	/admin/category/update/input	AdminCategoryUpdateController
3-4-2-3	カテゴリ変更確認画面	/admin/category/update/check	AdminCategoryUpdateController
3-4-3-3	カテゴリ変更完了画面	/admin/category/update/complete	AdminCategoryUpdateController
3-5-1-3	カテゴリ削除確認画面	/admin/category/delete/check	AdminCategoryDeleteController
3-5-2-3	カテゴリ削除完了画面	/admin/category/delete/complete	AdminCategoryDeleteController
4-1-1-2	買い物かご画面	/client/basket/list	ClientBasketController
	買い物かご商品追加	/client/basket/add	ClientBasketController
	買い物かご商品削除	/client/basket/delete	ClientBasketController
	買い物かご商品全削除	/client/basket/allDelete	ClientBasketController

画面 ID	画面名	URL パターン	ファイル名
5-1-1-2	注文一覧画面	/client/order/list	ClientOrderShowController
5-1-1-3	注文一覧画面	/admin/order/list	ClientOrderShowController
5-2-1-2	注文詳細画面	/client/order/detail/{id}	ClientOrderShowController
5-2-1-3	注文詳細画面	/admin/order/detail/{id}	AdminOrderSHowController
5-3-1-2	届け先入力画面	/client/order/address/input	ClientOrderRegistController
5-3-2-2	支払方法選択画面	/client/order/payment/input	ClientOrderRegistController
5-3-3-2	注文内容確認画面	/client/order/check	ClientOrderRegistController
5-3-4-2	注文完了画面	/client/order/complete	ClientOrderRegistController
6-1-1-3 6-1-1-4	会員一覧画面	/admin/user/list	AdminUserShowController
6-2-1-2	会員詳細画面	/client/user/detail	ClientUserShowController
6-2-1-3 6-2-1-4	会員詳細画面	/admin/user/detail/{id}	AdminUserShowController
6-3-1-1	会員登録入力画面	/client/user/regist/input	ClientUserRegistController
6-3-2-1	会員登録確認画面	/client/user/regist/check	ClientUserRegistController
6-3-3-2	会員登録完了画面	/client/user/regist/complete	ClientUserRegistController
6-3-1-3 6-3-1-4	会員登録入力画面	/admin/user/regist/input	AdminUserRegistController
6-3-2-3 6-3-2-4	会員登録確認画面	/admin/user/regist/check	AdminUserRegistController
6-3-3-3 6-3-3-4	会員登録完了画面	/admin/user/regist/complete	AdminUserRegistController
6-4-1-2	会員変更入力画面	/client/user/update/input	ClientUserUpdateIController
6-4-2-2	会員変更確認画面	/client/user/update/check	ClientUserUpdateIController
6-4-3-2	会員変更完了画面	/client/user/update/complete	ClientUserUpdateIController
6-4-1-3 6-4-1-4	会員変更入力画面	/admin/user/update/input	AdminUserUpdateController
6-4-2-3 6-4-2-4	会員変更確認画面	/admin/user/update/check	AdminUserUpdateController
6-4-3-3 6-4-3-4	会員変更完了画面	/admin/user/update/complete	AdminUserUpdateController
6-5-1-2	会員削除確認画面	/client/user/delete/check	ClientUserDeleteController
6-5-2-1	会員削除完了画面	/client/user/delete/complete	ClientUserDeleteController
6-5-1-3 6-5-1-4	会員削除確認画面	/admin/user/delete/check	AdminUserDeleteController
6-5-2-3 6-5-1-4	会員削除完了画面	/admin/user/delete/complete	AdminUserDeleteController
-	エラー画面	/syserror	ErrorController

2-4 テンプレートファイル構成

・ユーザインターフェースは、Thymeleaf のテンプレートを利用する
・テンプレート ファイルは 、/resources/template 配下に置く
・画面は、いずれかのレイアウトパターン(詳細は、後述)を利用する
・/resources/template 配下のパス名・ファイル名を下記に示す
・網掛け部分は、現行システムで実現済み

画面 ID	画面名	テンプレートパス	ファイル名	レイアウトパターン
共通部品	ヘッダ	/common	head.html	-
共通部品	ナビゲーションバー	/common	menu.html	-
共通部品	サイドバー	/common	sidebar.html	-
共通部品	フッタ	/common	footer.html	-
共通部品	レイアウトパターン 1	/common	layout_3block.html	-
共通部品	レイアウトパターン 2	/common	layout_4block.html	-
共通部品	レイアウトパターン 3	/common	layout_5block.html	-
-	エラー画面	/	error.html	パターン 1
1-1-1-1	ログイン画面	/login	login.html	パターン 1
1-3-1-3 1-3-1-4	管理者用メニュー画面	/admin/menu	admin_menu.html	パターン 1
2-1-1-1 2-1-1-2	トップ画面	/	index.html	パターン 3
2-1-2-1 2-1-2-2	商品一覧画面	/client/item	list.html	パターン 3
2-1-1-3	商品一覧画面	/admin/item	list.html	パターン 2
2-2-1-1 2-2-1-2	商品詳細画面	/client/item	detail.html	パターン 3
2-2-1-3	商品詳細画面	/admin/item	detail.html	パターン 2
2-3-1-3	商品登録入力画面	/admin/item	regist_input.html	パターン 2
2-3-2-3	商品登録確認画面	/admin/item	regist_check.html	パターン 2
2-3-3-3	商品登録完了画面	/admin/item	regist_complete.html	パターン 2
2-4-1-3	商品変更入力画面	/admin/item	update_input.html	パターン 2
2-4-2-3	商品変更確認画面	/admin/item	update_check.html	パターン 2
2-4-3-3	商品変更完了画面	/admin/item	update_complete.html	パターン 2
2-5-1-3	商品削除確認画面	/admin/item	delete_check.html	パターン 2
2-5-2-3	商品削除確認画面	/admin/item	delete_complete.html	パターン 2

画面 ID	画面名	テンプレートパス	ファイル名	レイアウトパターン
3-1-1-3	カテゴリー覧画面	/admin/category	list.html	パターン 2
3-2-1-3	カテゴリ詳細画面	/admin/category	detail.html	パターン 2
3-3-1-3	カテゴリ登録入力画面	/admin/category	regist_input.html	パターン 2
3-3-2-3	カテゴリ登録確認画面	/admin/category	regist_check.html	パターン 2
3-3-3-3	カテゴリ登録完了画面	/admin/category	regist_complete.html	パターン 2
3-4-1-3	カテゴリ変更入力画面	/admin/category	update_input.html	パターン 2
3-4-2-3	カテゴリ変更確認画面	/admin/category	update_check.html	パターン 2
3-4-3-3	カテゴリ変更完了画面	/admin/category	update_complete.html	パターン 2
3-5-1-3	カテゴリ削除確認画面	/admin/category	delete_check.html	パターン 2
3-5-2-3	カテゴリ削除完了画面	/admin/category	delete_complete.html	パターン 2
4-1-1-2	買い物かご画面	/client/basket	list.html	パターン 3
5-1-1-2	注文一覧画面	/client/order	list.html	パターン 3
5-1-1-3	注文一覧画面	/admin/order	list.html	パターン 2
5-2-1-2	注文詳細画面	/client/order	detail.html	パターン 3
5-2-1-3	注文詳細画面	/admin/order	detail.html	パターン 2
5-3-1-2	届け先入力画面	/client/order	address_input.html	パターン 3
5-3-2-2	支払方法選択画面	/client/order	payment_input.html	パターン 3
5-3-3-2	注文内容確認画面	/client/order	check.html	パターン 3
5-3-4-2	注文完了画面	/client/order	complete.html	パターン 3
6-1-1-3 6-1-1-4	会員一覧画面	/admin/user	list.html	パターン 2
6-2-1-2	会員詳細画面	/client/user	detail.html	パターン 3
6-2-1-3 6-2-1-4	会員詳細画面	/admin/user	detail.html	パターン 2
6-3-1-1	会員登録入力画面	/client/user	regist_input.html	パターン 3
6-3-2-1	会員登録確認画面	/client/user	regist_check.html	パターン 3
6-3-3-2	会員登録完了画面	/client/user	regist_complete.html	パターン 3
6-3-1-3 6-3-1-4	会員登録入力画面	/admin/user	regist_input.html	パターン 2
6-3-2-3 6-3-2-4	会員登録確認画面	/admin/user	regist_check.html	パターン 2
6-3-3-3 6-3-3-4	会員登録完了画面	/admin/user	regist_complete.html	パターン 2

画面 ID	画面名	テンプレートパス	ファイル名	レイアウトパターン
6-4-1-2	会員変更入力画面	/client/user	update_input.html	パターン３
6-4-2-2	会員変更確認画面	/client/user	update_check.html	パターン３
6-4-3-2	会員変更完了画面	/client/user	update_complete.html	パターン３
6-4-1-3 6-4-1-4	会員変更入力画面	/admin/user	update_input.html	パターン２
6-4-2-3 6-4-2-4	会員変更確認画面	/admin/user	update_check.html	パターン２
6-4-3-3 6-4-3-4	会員変更完了画面	/admin/user	update_complete.html	パターン２
6-5-1-2	会員削除確認画面	/client/user	delete_check.html	パターン３
6-5-2-1	会員削除完了画面	/client/user	delete_complete.html	パターン３
6-5-1-3 6-5-1-4	会員削除確認画面	/admin/user	delete_check.html	パターン２
6-5-2-3 6-5-1-4	会員削除完了画面	/admin/user	delete_complete.html	パターン２
-	エラー画面	/	error.html	パターン１

2 - 5 URL 一覧

・http://localhost:55000/shared_shop/に続くパス名の一覧を下記に示す
・テンプレートの form タグの action 属性や a タグの href 属性に指定する
・コントローラクラスの@RequestMapping に指定する URL パターンとパス名を紐づける
・該当するボタンやリンクに付与される URL の形式をあらわしている
・網掛け部分は、現行システムで実現済み機能

「xxxx」:固定値、{xxxx}:可変値

画面 ID	画面名・部品名	テンプレート内指定箇所	パス名	メソッド
–	ヘッダ	「ログイン」リンク	/login	GET
–	ヘッダ	「新規会員登録」リンク	/client/user/regist/input/init	GET
–	ヘッダ(一般会員)	{会員氏名} リンク	/client/user/detail	GET
–	ヘッダ	「ログアウト」リンク	/logout	GET
–	ヘッダ (運用管理者・システム管理者)	{会員氏名} リンク	/admin/user/detail/{id}	GET
–	ナビゲーションバー	「トップ」リンク	/	GET
–	ナビゲーションバー	「商品一覧」リンク	/client/item/list/1	GET
–	ナビゲーションバー	「買い物かご」リンク	/client/basket/list	GET
–	ナビゲーションバー	「注文一覧」リンク	/client/order/list	GET
–	サイドバー (非会員・一般会員)	カテゴリ別検索「検索」ボタン	/client/item/list/{sortType}?categoryId={カテゴリ ID}	GET
–	サイドバー (運用管理者・システム管理者)	「会員一覧表示」リンク	/admin/user/list	GET
–	サイドバー(運用管理者)	「注文一覧表示」リンク	/admin/order/list	GET
–	サイドバー(運用管理者)	「商品一覧表示」リンク	/admin/item/list	GET
–	サイドバー(運用管理者)	「カテゴリー一覧表示」リンク	/admin/category/list	GET
1-1-1-1	ログイン画面	「ログイン」ボタン	/login	POST
1-1-1-1	ログイン画面	「トップへ戻る」リンク	/	GET
1-3-1-3 1-3-1-4	管理者用メニュー画面	「会員一覧表示」リンク	/admin/user/list	GET
1-3-1-3	管理者用メニュー画面	「注文一覧表示」リンク	/admin/order/list	GET
1-3-1-3	管理者用メニュー画面	「商品一覧表示」リンク	/admin/item/list	GET
1-3-1-3	管理者用メニュー画面	「カテゴリー一覧表示」リンク	/admin/category/list	GET

「xxxx」:固定値、{xxxx}:可変値

画面 ID	画面名	テンプレート内指定箇所	パス名	メソッド
2-1-1-1 2-1-1-2	トップ画面	{商品画像}リンク	/client/item/detail/{id}	GET
2-2-1-1 2-2-1-2	商品一覧画面	「新着順」「売れ筋順」リンク	/client/item/list/{sortType}?categoryId={カテゴリ ID}	GET
2-1-2-1 2-1-2-2	商品一覧画面	{商品名}リンク	/client/item/detail/{id}	GET
2-1-1-3	商品一覧画面	「商品新規登録」ボタン	/admin/item/regist/input	POST
2-1-1-3	商品一覧画面	{商品名}リンク	/admin/item/detail/{id}	GET
2-1-1-3	商品一覧画面	「変更」ボタン	/admin/item/update/input/{id}	POST
2-1-1-3	商品一覧画面	「削除」ボタン	/admin/item/delete/check/{id}	POST
2-2-1-1 2-2-1-2	商品詳細画面	「買い物かごへ入れる」ボタン	/client/basket/add	POST
2-2-1-1 2-2-1-2	商品詳細画面	「戻る」ボタン	/client/item/list/1	POST
2-2-1-3	商品詳細画面	「戻る」ボタン	/admin/item/list	POST
2-2-1-3	商品詳細画面	「変更」ボタン	/admin/item/update/input/{id}	POST
2-2-1-3	商品詳細画面	「削除」ボタン	/admin/item/delete/check/{id}	POST
2-3-1-3	商品登録入力画面	「確認」ボタン	/admin/item/regist/check	POST
2-3-1-3	商品登録入力画面	「戻る」ボタン	/admin/item/list	POST
2-3-2-3	商品登録確認画面	「登録」ボタン	/admin/item/regist/complete	POST
2-3-2-3	商品登録確認画面	「戻る」ボタン	/admin/item/regist/input	POST
2-3-3-3	商品登録完了画面	「商品一覧へ戻る」リンク	/admin/item/list	GET
2-4-1-3	商品変更入力画面	「確認」ボタン	/admin/item/update/check	POST
2-4-1-3	商品変更入力画面	「戻る」ボタン	/admin/item/detail/{id}	POST
2-4-2-3	商品変更確認画面	「登録」ボタン	/admin/item/update/complete	POST
2-4-2-3	商品変更確認画面	「戻る」ボタン	/admin/item/update/input/{id}	POST
2-4-3-3	商品変更完了画面	「商品一覧へ戻る」リンク	/admin/item/list	GET
2-5-1-3	商品削除確認画面	「確認」ボタン	/admin/item/delete/complete	POST
2-5-1-3	商品削除確認画面	「戻る」ボタン	/admin/item/detail/{id}	POST
2-5-2-3	商品削除完了画面	「商品一覧へ戻る」リンク	/admin/item/list	GET

「xxxx」:固定値、{xxxx}:可変値

画面 ID	画面名	テンプレート内指定箇所	パス名	メソッド
3-1-1-3	カテゴリ一覧画面	「カテゴリ新規登録」ボタン	/admin/category/regist/input	GET
3-1-1-3	カテゴリ一覧画面	{カテゴリ名}リンク	/admin/category/detail/{id}	GET
3-1-1-3	カテゴリ一覧画面	「変更」ボタン	/admin/category/update/input/{id}	POST
3-1-1-3	カテゴリ一覧画面	「削除」ボタン	/admin/category/delete/check/{id}	POST
3-2-1-3	カテゴリ詳細画面	「戻る」ボタン	/admin/category/list	POST
3-2-1-3	カテゴリ詳細画面	「変更」ボタン	/admin/category/update/input/{id}	POST
3-2-1-3	カテゴリ詳細画面	「削除」ボタン	/admin/category/delete/check/{id}	POST
3-3-1-3	カテゴリ登録入力画面	「確認」ボタン	/admin/category/regist/check	POST
3-3-1-3	カテゴリ登録入力画面	「戻る」ボタン	/admin/category/list	POST
3-3-2-3	カテゴリ登録確認画面	「登録」ボタン	/admin/category/regist/complete	POST
3-3-2-3	カテゴリ登録確認画面	「戻る」ボタン	/admin/category/regist/input	POST
3-3-2-3	カテゴリ登録確認画面	「登録」ボタン	/admin/category/regist/complete	POST
3-3-3-3	カテゴリ登録完了画面	「カテゴリ一覧へ戻る」リンク	/admin/category/list	GET
3-4-1-3	カテゴリ変更入力画面	「確認」ボタン	/admin/category/update/check	POST
3-4-1-3	カテゴリ変更入力画面	「戻る」ボタン	/admin/category/detail/{id}	POST
3-4-2-3	カテゴリ変更確認画面	「登録」ボタン	/admin/category/update/complete	POST
3-4-2-3	カテゴリ変更確認画面	「戻る」ボタン	/admin/category/update/input/{id}	POST
3-4-3-3	カテゴリ変更完了画面	「カテゴリ一覧へ戻る」リンク	/admin/category/list	GET
3-5-1-3	カテゴリ削除確認画面	「削除」ボタン	/admin/category/delete/complete	POST
3-5-1-3	カテゴリ削除確認画面	「戻る」ボタン	/admin/category/detail/{id}	POST
3-5-2-3	カテゴリ削除完了画面	「カテゴリ一覧へ戻る」リンク	/admin/category/list	GET
4-1-1-2	買い物かご画面	「削除」ボタン	/client/basket/delete	POST
4-1-1-2	買い物かご画面	「ご注文のお手続き」ボタン	/client/order/address/input	POST
4-1-1-2	買い物かご画面	「買い物かごを空にする」ボタン	/client/basket/allDelete	POST
5-1-1-2	注文一覧画面	{注文日時}リンク	/client/order/detail/{id}	GET
5-1-1-3	注文一覧画面	{会員氏名}リンク	/admin/order/detail/{id}	GET
5-2-1-2	注文詳細画面	「戻る」ボタン	/client/order/list	POST
5-2-1-3	注文詳細画面	「戻る」ボタン	/admin/order/list	POST
5-3-1-2	届け先入力画面	「次へ」ボタン	/client/order/payment/input	POST
5-3-1-2	届け先入力画面	「戻る」ボタン	/client/basket/list	POST
5-3-2-2	支払方法選択画面	「次へ」ボタン	/client/order/check	POST
5-3-2-2	支払方法選択画面	「戻る」ボタン	/client/order/address/input	POST
5-3-3-2	注文内容確認画面	「ご注文の確定」ボタン	/client/order/complete	POST
5-3-3-2	注文内容確認画面	「戻る」ボタン	/client/order/payment/back	POST
5-3-4-2	注文完了画面	「トップ画面へ」リンク	/	GET

「xxxx」:固定値、{xxxx}:可変値

画面 ID	画面名	テンプレート内指定箇所	パス名	メソッド
6-1-1-3 6-1-1-4	会員一覧表示画面	「新規会員登録」ボタン	/admin/user/regist/input	POST
6-1-1-3 6-1-1-4	会員一覧画面	{会員氏名}リンク	/admin/user/detail/{id}	GET
6-1-1-3 6-1-1-4	会員一覧画面	「変更」ボタン	/admin/user/update/input/{id}	POST
6-1-1-3 6-1-1-4	会員一覧画面	「削除」ボタン	/admin/user/delete/check/{id}	POST
6-2-1-2	会員詳細画面	「戻る」ボタン	/	POST
6-2-1-2	会員詳細画面	「変更」ボタン	/client/user/update/input	POST
6-2-1-2	会員詳細画面	「退会」ボタン	/client/user/delete/check	POST
6-2-1-3 6-2-1-4	会員詳細画面	「戻る」ボタン	/admin/user/list	POST
6-2-1-3 6-2-1-4	会員詳細画面	「変更」ボタン	/admin/user/update/input/{id}	POST
6-2-1-3 6-2-1-4	会員詳細画面	「削除」ボタン	/admin/user/delete/check/{id}	POST
6-3-1-1	会員登録入力画面	「確認」ボタン	/client/user/regist/check	POST
6-3-1-1	会員登録入力画面	「戻る」ボタン	/	POST
6-3-2-1	会員登録確認画面	「登録」ボタン	/client/user/regist/complete	POST
6-3-2-1	会員登録確認画面	「戻る」ボタン	/client/user/regist/input	POST
6-3-3-2	会員登録完了画面	「トップへ戻る」リンク	/	GET
6-3-1-3 6-3-1-4	会員登録入力画面	「確認」ボタン	/admin/user/regist/check	POST
6-3-1-3 6-3-1-4	会員登録入力画面	「戻る」ボタン	/admin/user/list	POST
6-3-2-3 6-3-2-4	会員登録確認画面	「登録」ボタン	/admin/user/regist/complete	POST
6-3-2-3 6-3-2-4	会員登録確認画面	「戻る」ボタン	/admin/user/regist/input	POST
6-3-3-3 6-3-3-4	会員登録完了画面	「会員一覧へ戻る」リンク	/admin/user/list	GET

「xxxx」:固定値、{xxxx}:可変値

画面 ID	画面名	テンプレート内指定箇所	パス名	メソッド
6-4-1-2	会員変更入力画面	「確認」ボタン	/client/user/update/check	POST
6-4-1-2	会員変更入力画面	「戻る」ボタン	/client/user/detail	POST
6-4-2-2	会員変更確認画面	「登録」ボタン	/client/user/update/complete	POST
6-4-2-2	会員変更確認画面	「戻る」ボタン	/client/user/update/input	POST
6-4-3-2	会員変更完了画面	「会員詳細へ戻る」リンク	/client/user/detail	GET
6-4-1-3 6-4-1-4	会員変更入力画面	「確認」ボタン	/admin/user/update/check	POST
6-4-1-3 6-4-1-4	会員変更入力画面	「戻る」ボタン	/admin/user/detail/{id}	POST
6-4-2-3 6-4-2-4	会員変更確認画面	「登録」ボタン	/admin/user/update/complete	POST
6-4-2-3 6-4-2-4	会員変更確認画面	「戻る」ボタン	/admin/user/update/input/{id}	POST
6-4-3-3 6-4-3-4	会員変更完了画面	「会員一覧へ戻る」リンク	/admin/user/list	GET
6-5-1-2	会員削除確認画面	「退会」ボタン	/client/user/delete/check	POST
6-5-1-2	会員削除確認画面	「戻る」ボタン	/client/user/detail	POST
6-5-2-1	会員削除完了画面	「トップへ戻る」リンク	/	GET
6-5-1-3 6-5-1-4	会員削除確認画面	「削除」ボタン	/admin/user/delete/complete	POST
6-5-1-3 6-5-1-4	会員削除確認画面	「戻る」ボタン	/admin/user/detail/{id}	POST
6-5-2-3 6-5-2-4	会員削除完了画面	「会員一覧へ戻る」リンク	/admin/user/list	GET
-	エラー画面	-	/syserror	GET

3 ユーザインターフェース設計

3 - 1 共通部品と画面パターン

Thymeleaf のレイアウト機能を利用し、別ファイルで作成した共通部品を埋め込む形式とする。

(1)共通部品

- ヘッダ　　ファイル名:/common/header.html
- ナビゲーションバー　　ファイル名:/common/menu.html
- サイドバー　　ファイル名:/common/sidebar.html
- フッタ　　ファイル名:/common/footer.html

(2)レイアウト

レイアウトは、以下の３つのパターンを用いる。

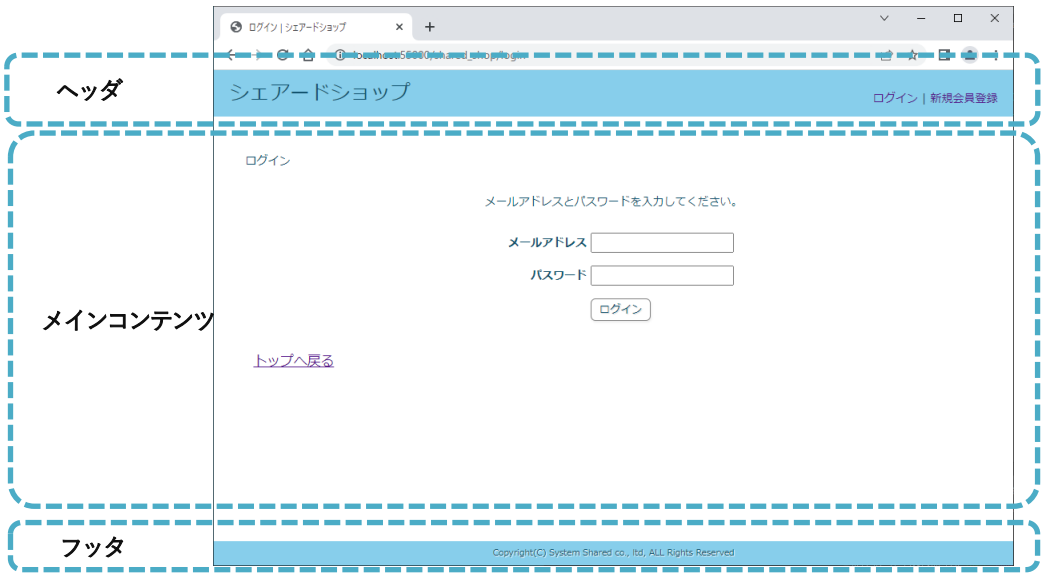
<レイアウトパターン 1>

・構成要素:ヘッダ、メインコンテンツ、フッタの３ブロック構成

・レイアウトファイル名: /common/layout_3block.html

・使用画面：

- 1-1-1-1 ログイン画面
- 1-3-1-3,1-3-1-4 管理者用メニュー画面
- エラー画面



<レイアウトパターン2>

・構成要素: ヘッダ、サイドバー、メインコンテンツ、フッタ

・レイアウトファイル名: /common/layout_4block.html

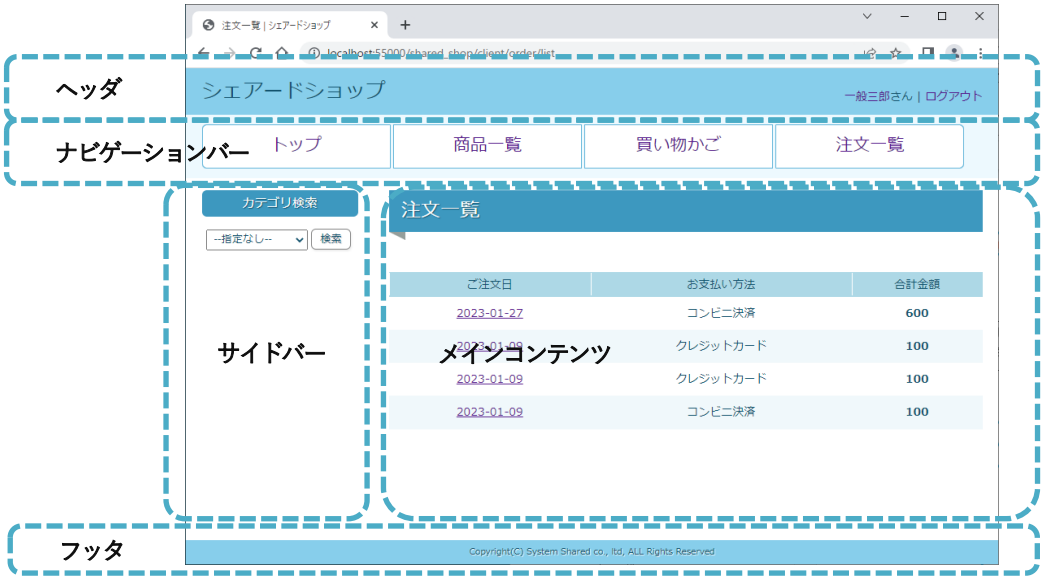
・使用画面：運用管理者、システム管理者用画面

<レイアウトパターン3>

・構成要素:ヘッダ、ナビゲーションバー、サイドバー、メインコンテンツ、フッタ

・レイアウトファイル名: /common/layout_5block.html

・使用画面：非会員、一般会員用画面



3-2入力値チェック

・入力値の条件と入力チェック用のアノテーションを下記に記載する。
・アノテーションは、javax.validation.constraints パッケージのアノテーションを利用する。
・以下3点については、独自アノテーションを利用する。
- ログイン認証@LoginCheck
- メールアドレス重複チェック@EmailCheck
- カテゴリ名重複チェック@CategoryCheck

(1)ログイン情報
入力画面:1-1-1-1 ログイン画面

項目名	形式	最小値/ 最小文字数	最大値/ 最大文字数	利用アノテーション(チェック内容)
メールアドレス	メールアドレス形式 (RFC2822 準拠)	1	129 @前後 64 文字	@NotBlank @Email
パスワード	半角英数字	8	16	@NotBlank @Size @Pattern
メールアドレスと パスワードの組合せ	-	-	-	その他:DB 問い合わせによる確認

(2)商品情報
入力画面:2-3-1-3 商品登録入力画面、2-4-1-3 商品変更入力画面

項目名	形式	最小値/ 最小文字数	最大値/ 最大文字数	利用アノテーション
商品名	半角文字、全角文字	1	100	@NotBlank @Size
価格	半角数字 (桁区切り文字なし)	0	9999999	@NotNull @Max
在庫数	半角数字	0	9999	@NotNull @Max
商品カテゴリ	選択式(半角数字)	1	99	-
説明文	半角文字、全角文字	0	400	@Size
商品画像	ファイル(jpg のみ)	0	1MB	-

(3)カテゴリ情報

入力画面:3-3-1-3 カテゴリ登録入力画面、3-4-1-3 カテゴリ変更入力画面

制限事項:カテゴリ情報の登録可能件数は、99 件(削除済みも含む)とする。

項目名	形式	最小値/ 最小文字数	最大値/ 最大文字数	利用アノテーション
カテゴリ名※	半角文字、全角文字	1	15	@NotBlank @Size
説明文	半角文字、全角文字	0	30	@Size

(4)注文情報

入力画面:5-3-1-2 届け先入力画、5-3-2-2 支払方法選択画面

項目名	形式	最小値/ 最小文字数	最大値/ 最大文字数	利用アノテーション
お届け先郵便番号	半角数字 ハイフン「-」なし	7	7	@NotBlank @Size @Pattern
お届け先住所	半角文字、全角文字	1	150	@NotBlank @Size
お届け先氏名	半角文字、全角文字	1	30	@NotBlank @Size
お届け先電話番号	半角数字 ハイフン「-」なし	10	11	@NotBlank @Size @Pattern
支払い方法	選択入力(半角数字)	1	5	-

⑸会員情報

入力画面:6-3-1-1、6-3-1-3、6-3-1-4 会員登録入力画面
6-4-1-2、6-4-1-3、6-4-1-4 会員変更入力画面

項目名	形式	最小値/ 最小文字数	最大値/ 最大文字数	利用アノテーション
メールアドレス	メールアドレス形式 (RFC2822 準拠)	1	129 @前後 64 文字	@NotBlank @Email
パスワード	半角英数字	8	16	@NotBlank @Size @Pattern
氏名	半角文字、全角文字	1	30	@NotBlank @Size
郵便番号	半角数字 ハイフン「-」なし	7	7	@NotBlank @Size @Pattern
住所	半角文字、全角文字	1	150	@NotBlank @Size
電話番号	半角数字 ハイフン「-」なし	10	11	@NotBlank @Size @Pattern
権限※	選択入力 (半角数字)	0	1	－

※権限 6-3-1-4 のみ、ラジオボタンで選択可能　0:システム管理者、1:運用管理者

4 プログラム構造設計

4 - 1 概要

拡張性、再利用性、セキュリティ侵害に対する堅牢性を考慮したプログラム構造とするために
下記の点を決定した。

- アクセス制御の方式
- 2 重送信防止を考慮した画面遷移が必要な処理の構造(Post-Redirect-Get 構造)
  - ログイン処理
  - 登録処理
  - 変更処理
  - 削除処理
  - 買い物かご
  - 注文登録処理
- 在庫数の取扱い
- エラー処理の構造
- 想定外の画面遷移への対処

4-2アクセス制御の方式

・利用者の権限に応じてアクセスできる機能を制限するために、フィルタを利用する。
・権限の判別は、以下の方法で行う
・セッションスコープにログイン情報があるか否か
- ログイン情報がない場合、非会員機能及び会員削除完了画面
- ログイン情報がある場合、権限は一般会員、運用管理者、システム管理者のいずれか

(1)権限一覧

機能名				権限 ○:利用可能 ×:利用できない			
大分類		小分類		未ログイン	一般会員	運用管理者	システム管理者
1	汎用	1	ログイン	○	－	－	－
		2	ログアウト	－	○	○	○
2	商品管理	1	商品一覧表示	○	○	○	×
		2	表示順変更(新着順)	○	○	×	×
		3	表示順変更(売れ筋順)	○	○	×	×
		4	カテゴリ別検索	○	○	×	×
		5	商品詳細表示	○	○	○	×
		6	商品登録	×	×	○	×
		7	商品変更	×	×	○	×
		8	商品削除	×	×	○	×
3	カテゴリ管理	1	カテゴリー覧表示	×	×	○	×
		2	カテゴリ詳細表示	×	×	○	×
		3	カテゴリ登録	×	×	○	×
		4	カテゴリ変更	×	×	○	×
		5	カテゴリ削除	×	×	○	×
4	買い物かご管理	1	商品一覧表示	×	○	×	×
		2	商品追加	×	○	×	×
		3	商品個数変更	×	○	×	×
		4	商品削除	×	○	×	×
5	注文管理	1	注文一覧表示	×	○※1	○※2	×
		2	注文詳細表示	×	○※1	○※2	×
		3	注文登録	×	○※1	×	×
6	会員管理	1	会員一覧表示	×	×	○※3	○※5
		2	会員詳細表示	×	○※1	○※3	○※5
		3	会員登録	○	×	○※4	○※6
		4	会員変更	×	○※1	○※4	○※6
		5	会員削除	×	○※1	○※4	○※6

※1 ログインしている会員自身の情報のみを扱うことができる
※2 全ての会員の注文情報を閲覧することができる
※3 一般会員、運用管理者の情報を閲覧できる
※4 運用管理者権限の会員を登録、変更、削除ができ、一般会員の変更、削除ができる
ただし、権限変更は不可とし、ログインしている運用管理者自身の場合、削除も不可
※5 一般会員、運用管理者、システム管理者の情報を閲覧できる
※6 一般会員、運用管理者、システム管理者権限の会員を登録、変更、削除ができる
ただし、ログインしているシステム管理者自身の削除および、権限変更は不可

(2)AdminAccountCheckFilter(運用管理者用フィルタクラス)の処理

・/admin を含む URL にアクセスされた時の処理
・下記の条件に一致した場合、ログイン画面にリダイレクトする。
- 条件 1:Session スコープにログイン情報がない
- 条件 2:Session スコープにあるログイン情報の権限が一般会員

(3)CustomerAccountCheckFilter(一般会員用フィルタクラス)の処理

・下記を含む URL にアクセスされた時の処理
- /
- /index.html
- /top
- /item/*
- /basket/*
- /order/*
- /user/update/*
- /user/delete/check

・下記の条件に一致した場合、ログイン画面にリダイレクトする。
- 条件 1:Session スコープにあるログイン情報の権限が一般会員以外

(4)SystemAdminAccountCheckFilter(システム管理者用フィルタ)の処理

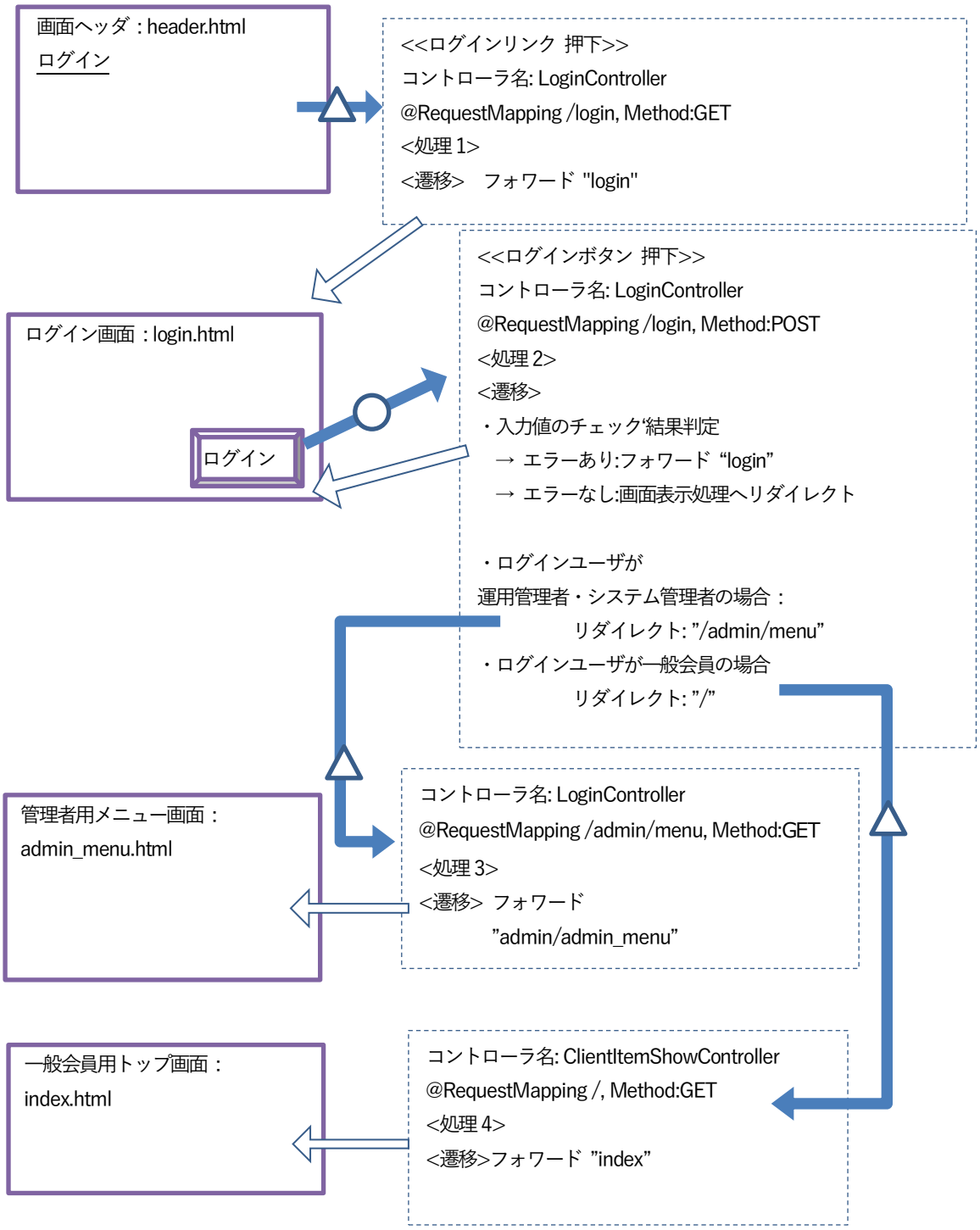
・/admin/order, /admin/item, /admin/category を含む URL にアクセスされた時の処理
・下記の条件に一致した場合、ログイン画面にリダイレクトする。
- 条件 1:Session スコープにログイン情報がない
- 条件 2:Session スコープにあるログイン情報の権限が運用管理者以外

4 - 3 ログイン処理の構造

(1)画面遷移とコントローラクラスとの関係

ログイン情報の多重送信を防止するため、Post-Redirect-Get パターンを用いる。

凡例: △:GET メソッド、〇:Post メソッド、処理説明は次項「コントローラクラス処理概要」に記載



(2)コントローラクラス処理概要

LoginController 処理

処理	マッピング, 処理内容
処理 1	(ログインリンク 押下時処理) path = "/login", method = RequestMethod.GET ・セッションスコープのログイン情報を破棄 ・ログイン画面表示 フォワード : “login”
処理 2	(ログインボタン 押下時処理) path = "/login", method = RequestMethod.POST ・リクエストスコープの入力エラーをチェック [エラーがある場合] ・セッションスコープのログイン情報を破棄 ・ログイン画面再表示 フォワード : “login” [エラーがない場合] ・セッションスコープのログイン情報から権限を取得 [一般利用者の場合] ・トップ画面表示処理へリダイレクト リダイレクト : “/” [運用管理者、システム管理者の場合] ・管理者メニュー画面表示処理へリダイレクト リダイレクト : “/admin/menu”
処理 3	(管理者メニュー画面表示処理‘) path = "/admin/menu", method = RequestMethod.GET ・管理者メニュー画面表示 フォワード : “admin/admin_menu”

ClientItemShowController 処理

処理	マッピング, 処理内容
処理 4	(ログインリンク 押下時処理) path = "/", method = RequestMethod.GET ・商品一覧表示の並び順を「売れ筋順」に初期化 ・注文情報'DB から、「売れ筋順」商品一覧情報を所得 ・「売れ筋順」商品一覧がなかった場合、 商品情報 DB から「新着順」商品一覧情報を所得 新着商品あり：一覧表示の並び順を「新着順」にする ・取得したログイン商品一覧情報を画面表示用一覧オブジェクトにコピー ・一覧表示の並び順と画面表示用一覧オブジェクトをリクエストオブジェクトに設定 ・トップ画面表示 フォワード：“index”

4 - 4 登録処理の構造

・下記の機能については、同様の構造とすることでメンテナンス性を高める
・処理は～RegistController クラスで行う
・なお、登録情報の多重送信を防止するため、Post-Redirect-Get パターンを用いる
・「戻る」ボタン押下時に入力値維持、再表示できるようにする。
・URL は戻り先 URL となるよう、リダイレクトを行う。
・入力内容はセッションスコープを利用して受渡する
・セッションスコープの該当する属性を、一覧画面表示時点もしくは入力画面表示時点で削除しておく

⑴対象機能

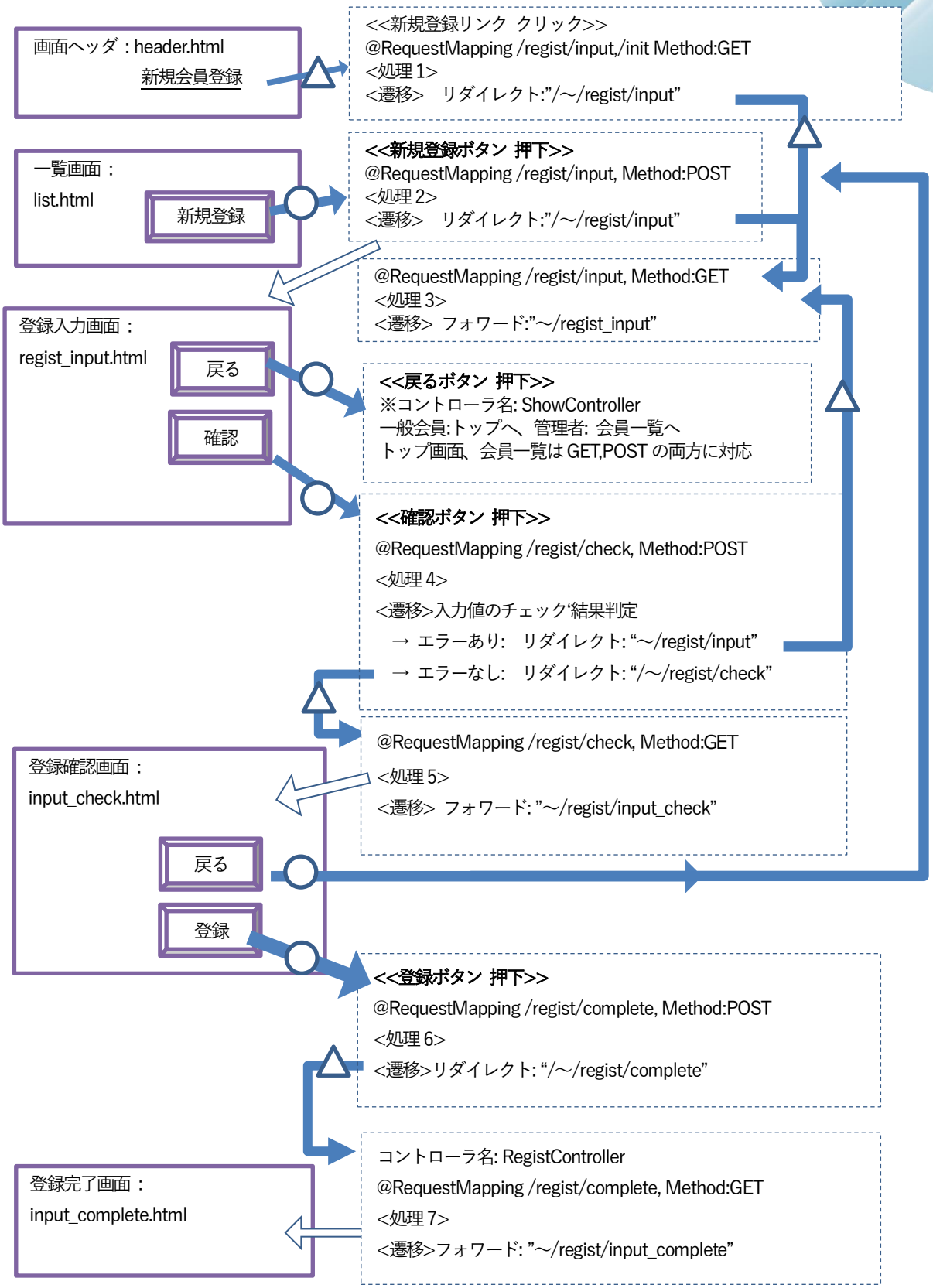
・2-3 商品登録 コントローラ : /admin/item/AdminItemRegistController
・3-3 カテゴリ登録 コントローラ : /admin/category/AdminCategoryRegistController
・6-3 会員登録
  - 一般会員登録 コントローラ : /client/user/ClientUserRegistController
  - 管理者情報登録 コントローラ : /admin/user/AdminUserRegistController

セッションスコープ内の入力値に関する属性を削除するコントローラ

・2-1 商品一覧 コントローラ : /admin/item/AdminItemShowController
・3-1 カテゴリ一覧 コントローラ : /admin/category/AdminCategoryShowController
・6-1 会員一覧 コントローラ : /admin/user/AdminUserShowController

(2)画面遷移とコントローラクラスとの関係

凡例 △:GET、○:POST、処理説明は次項「コントローラクラス処理概要」に記載



(3)コントローラクラス処理概要

～RegistController 処理

処理	マッピング, 処理内容
処理1	(新規登録リンク クリック時処理) path = "/～/regist/input/init", method = RequestMethod.GET ・入力フォーム情報を新規生成 ・入力フォーム情報をセッションスコープに保存 ・登録入力画面表示処理にリダイレクト リダイレクト："/～/regist/input"
処理2	(新規登録ボタン 押下時処理)、(確認画面-戻るボタン 押下時処理) path = "/～/regist/input", method = RequestMethod.POST ・セッションスコープに入力フォーム情報があるかを確認、なければ下記の処理を実施 ・入力フォーム情報を新規生成 ・処理の内容に応じて、必要な情報を入力フォーム情報にセット ・入力フォーム情報をセッションスコープに保存 ・登録入力画面表示処理にリダイレクト リダイレクト："/～/regist/input"
処理3	(登録画面表示処理) path = "/～/regist/input", method = RequestMethod.GET ・セッションスコープから入力フォーム情報を取得 ・入力フォーム情報をリクエストスコープに設定 ・セッションスコープに入力エラー情報がある場合 - 取得したエラー情報をリクエストスコープに設定 - セッションスコープから、エラー情報を削除 ・登録画面表示 フォワード: "～/regist_input"
処理4	(確認ボタン 押下時処理) path = "/～/regist/input/check", method = RequestMethod.POST ・セッションスコープからフォーム情報を取得 ・画面から入力された入力フォームを、セッションスコープに入力フォーム情報として保存 ・入力フォーム情報に不足がある場合、セッションスコープから取得した値をセット ・BindingResult オブジェクトに入力エラー情報がある場合 - 入力エラー情報と入力フォーム情報を設定 - 登録入力画面表示処理にリダイレクト リダイレクト："/～/regist/input" ・入力エラーがない場合 - 登録確認画面表示処理にリダイレクト リダイレクト："/～/regist/check"

処理	マッピング, 処理内容
処理5	(登録確認画面表示処理) path = "/~/regist/check", method = RequestMethod.GET ・セッションスコープから入力フォーム情報を取得 ・入力フォーム情報をリクエストスコープに設定 ・登録確認画面表示 フォワード :“~/regist_check”
処理6	(登録ボタン 押下時処理) path = "/~/regist/complete", method = RequestMethod.POST ・セッションスコープから入力フォーム情報を取得 ・入力フォーム情報を元に DB 登録用エンティティオブジェクトを生成 ・DB 登録実施 ・セッションスコープの入力フォーム情報削除 ・処理内容に応じて、セッションスコープの内容を書き換える - 未ログインでの会員登録の場合、セッションスコープに会員情報をセットしログイン状態にする カテゴリ登録の場合、セッションスコープのカテゴリ一覧を更新 ・登録完了画面表示処理にリダイレクト リダイレクト :"/~/regist/complete"
処理7	(登録完了画面表示処理) path = "/~/regist/complete", method = RequestMethod.GET ・登録完了画面表示 フォワード :“~/regist_complete”

4 - 5 変更処理の構造

・更新情報の多重送信を防止するため、Post-Redirect-Get パターンを用いる
・処理は～UpdateController クラスで行う
・「戻る」ボタン押下時に入力値維持、再表示できるようにする
・URL は戻り先 URL となるよう、リダイレクトを行う
・入力内容はセッションスコープを利用して受渡する
・セッションスコープの該当する属性を、一覧画面、詳細画面表示時点で削除しておく

(1)対象機能

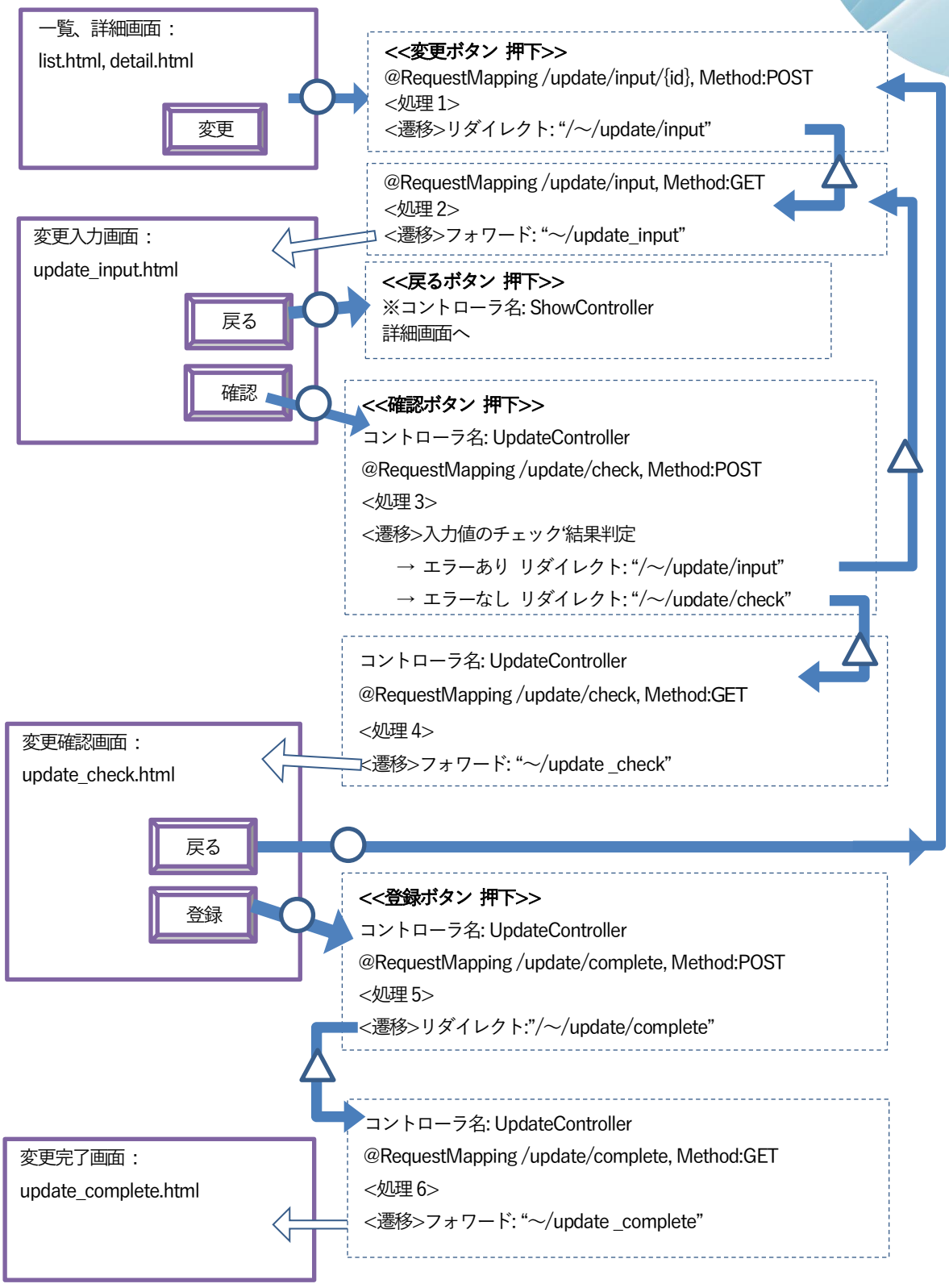
・2-8 商品情報変更 コントローラ : /admin/item/AdminItemUpdateController
・3-4 カテゴリ情報変更 コントローラ : /admin/category/AdminCategoryUpdateController
・6-4 会員情報変更
  - 一般会員変更 コントローラ : /client/user/ClientUserUpdateController
  - 会員情報変更 コントローラ : /admin/user/AdminUserUpdateController

セッションスコープ内の入力値に関する属性を削除するコントローラ

・2-1 商品一覧 コントローラ : /admin/item/AdminItemShowController
・3-1 カテゴリ一覧 コントローラ : /admin/category/AdminCategoryShowController
・6-1 会員一覧 コントローラ : /admin/user/AdminUserShowController
・2-2 商品詳細 コントローラ : /admin/item/AdminItemShowController
・3-2 カテゴリ詳細 コントローラ : /admin/category/AdminCategoryShowController
・6-2 会員詳細 コントローラ
  - 一般会員変更 コントローラ : /client/user/ClientUserShowController
  - 会員情報変更 コントローラ : /admin/user/AdminUserShowController

(2)画面遷移とコントローラクラスとの関係

凡例 △:GET、○:POST、処理説明は次項「コントローラクラス処理概要」に記載



(3)コントローラクラス処理概要

～UpdateController 処理

処理	マッピング, 処理内容
処理1	(変更ボタン 押下時処理) 、(確認画面-戻るボタン 押下時処理) path = "/～/update/input/{id}", method = RequestMethod.POST ・セッションスコープに入力フォーム情報があるかを確認、なければ下記の処理を実施 ・パスに指定の ID を条件に変更対象のデータを DB から取得 ・取得データを元に入力画面初期表示用の入力フォーム情報を新規生成 ・入力フォーム情報をセッションスコープに保存 ・変更入力画面表示処理へリダイレクト リダイレクト: "/～/update/input"
処理2	(変更入力画面表示処理) path = "/～/update/input", method = RequestMethod.GET ・セッションスコープから入力フォーム情報を取得 ・入力フォーム情報をリクエストスコープに設定 ・セッションスコープに入力エラー情報がある場合 - 取得した入力エラー情報をリクエストスコープに設定 - セッションスコープから、入力エラー情報を削除 ・変更入力画面表示 フォワード :“～/update_input”
処理3	(確認ボタン 押下時処理) path = "/～/update/check", method = RequestMethod.POST ・セッションスコープからフォーム情報を取得 ・画面から入力された入力フォームを、セッションスコープに入力フォーム情報として保存 ・入力フォーム情報に不足がある場合、セッションスコープから取得した値をセット ・BindingResult オブジェクトに入力エラー情報がある場合 - 入力エラー情報をセッションスコープに設定 - 変更入力画面表示処理にリダイレクト リダイレクト :“～/update/input” ・入力エラーがない場合 - 変更確認画面表示処理にリダイレクト リダイレクト :“/～/update/check”
処理4	(変更確認画面表示処理) path = "/～/update/check", method = RequestMethod.GET ・セッションスコープから入力フォーム情報を取得 ・入力フォーム情報をリクエストスコープに設定 ・登録確認画面表示 フォワード :“～/update_check”

処理	マッピング, 処理内容
処理5	(登録ボタン 押下時処理) path = "/～/update/complete", method = RequestMethod.POST ・セッションスコープから入力フォーム情報を取得 ・入力フォーム情報を元に DB 登録用エンティティオブジェクトを生成 ・DB 更新実施 ・セッションスコープの入力フォーム情報削除 ・処理内容に応じて、セッションスコープの内容を書き換える - ログインユーザの会員変更の場合、セッションスコープの会員情報を更新 - カテゴリ登録の場合、セッションスコープのカテゴリ一覧を更新 ・変更完了画面表示処理にリダイレクト リダイレクト :"/～/update/complete"
処理6	(変更完了画面表示処理) path = "/～/update/complete", method = RequestMethod.GET ・登録完了画面表示 フォワード :“～/update_complete”

4 - 6 削除処理の構造

下記の機能については、同様の構造とすることでメンテナンス性を高める。

・処理は～DeleteController クラスで行う
・入力内容はセッションスコープを利用して受渡する
・セッションスコープの該当する属性を、一覧画面、詳細画面表示時点で削除しておく

(1)対象機能

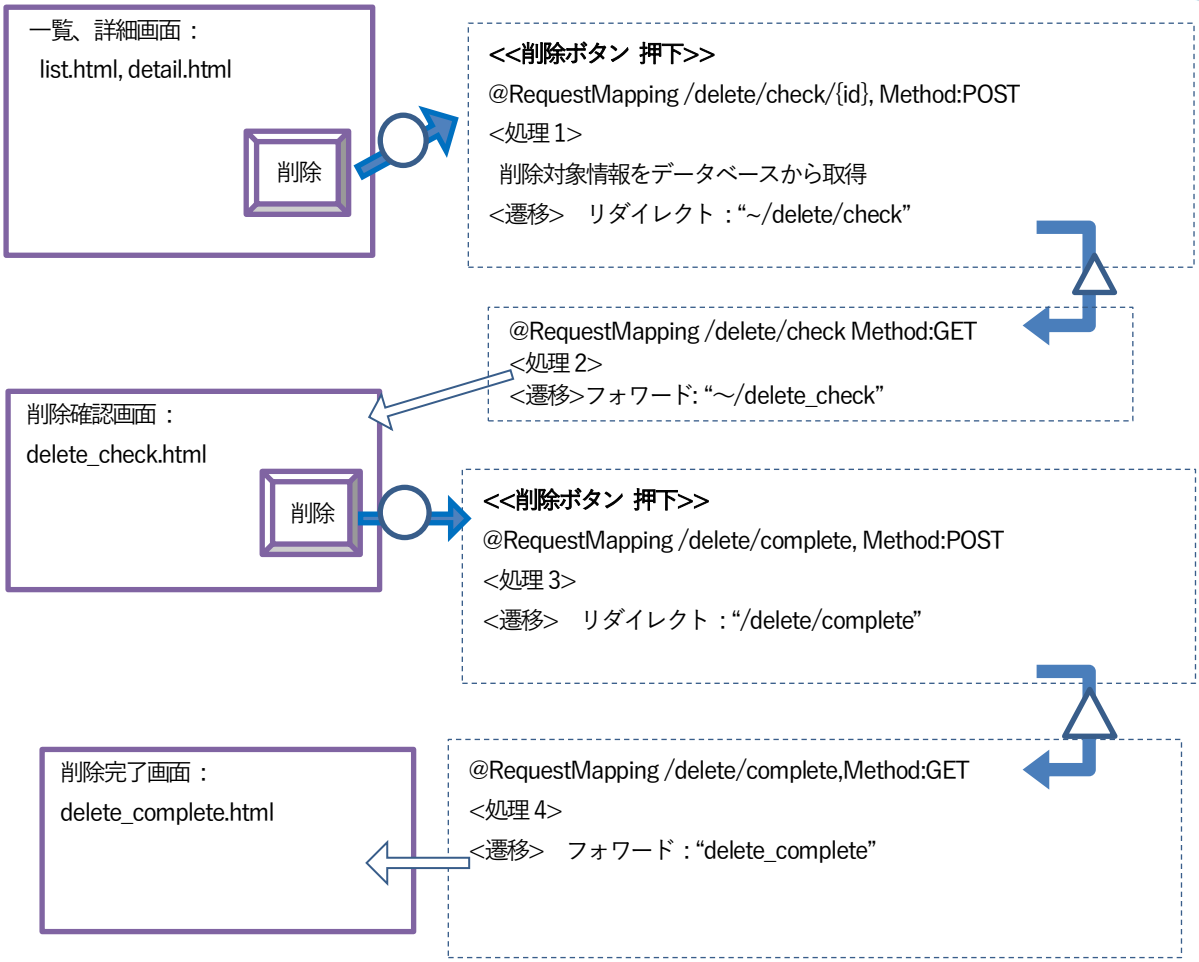
・2-9 商品削除 コントローラ：/admin/item/AdminItemDeleteController
・3-5 カテゴリ削除 コントローラ：/admin/category/AdminCategoryDeleteController
・6-5 会員削除
  - 一般会員削除 コントローラ：/client/user/ClientUserDeleteController
  - 会員情報削除 コントローラ：/admin/user/AdminUserDeleteController

セッションスコープ内の入力値に関する属性を削除するコントローラ

・2-1 商品一覧 コントローラ：/admin/item/AdminItemShowController
・3-1 カテゴリ一覧 コントローラ：/admin/category/AdminCategoryShowController
・6-1 会員一覧 コントローラ：/admin/user/AdminUserShowController
・2-2 商品詳細 コントローラ：/admin/item/AdminItemShowController
・3-2 カテゴリ詳細 コントローラ：/admin/category/AdminCategoryShowController
・6-2 会員詳細 コントローラ
  - 一般会員変更 コントローラ：/client/user/ClientUserShowController
  - 会員情報変更 コントローラ：/admin/user/AdminUserShowController

(2)画面遷移とコントローラクラスのメソッドとの関係

凡例　△:GET、○:POST、処理説明は次項「コントローラクラス処理概要」に記載



(3)コントローラクラス処理概要

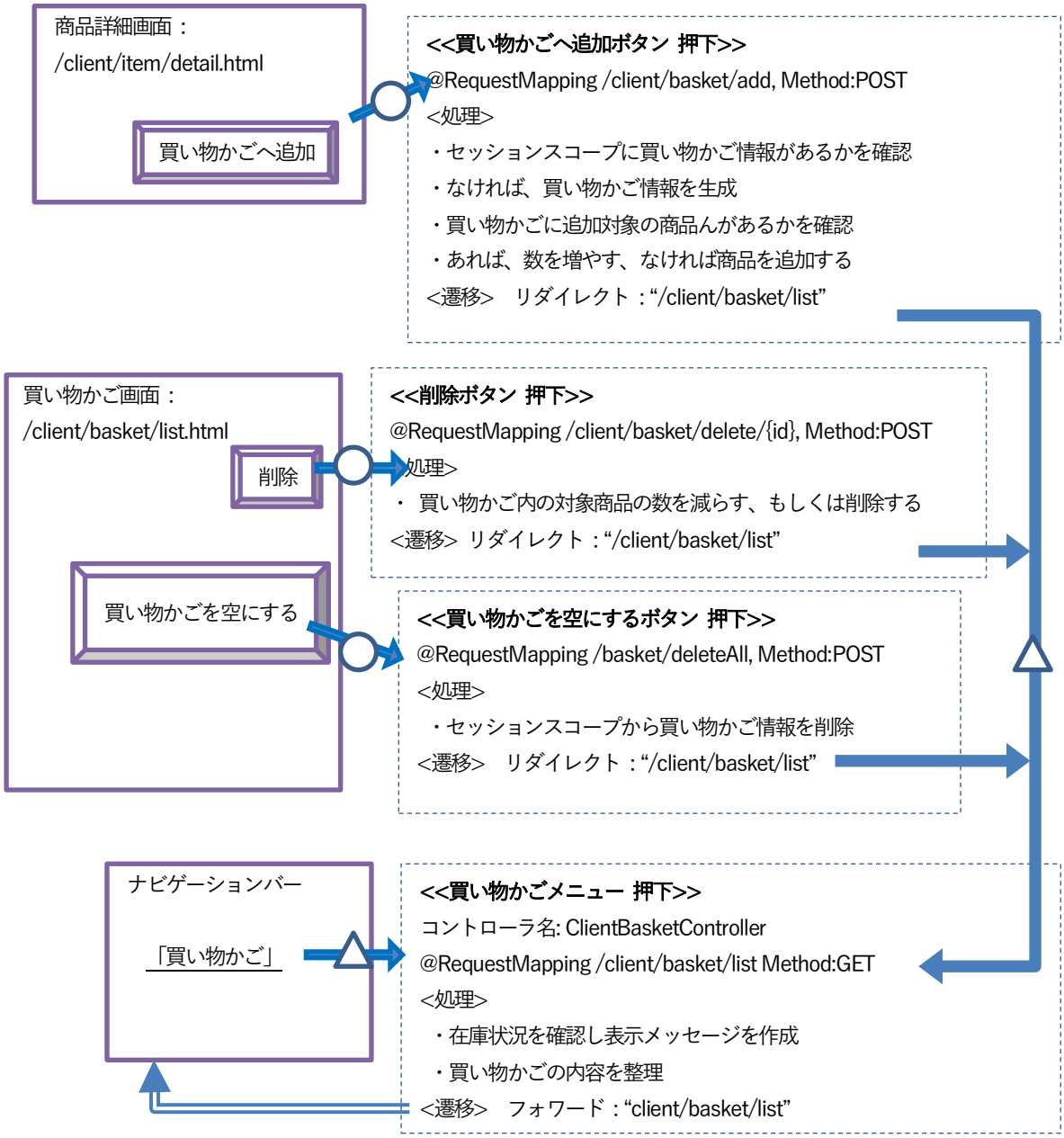
～DeleteController 処理

処理	マッピング, 処理内容
処理1	(削除ボタン 押下時処理) path = "/～/delete/input/{id}", method = RequestMethod.POST ・パスに指定の ID を条件に削除対象のデータを DB から取得 ・取得データを元に削除確認画面表示用の入力フォーム情報を新規生成 ・入力フォーム情報をセッションスコープに保存 ・削除確認画面表示処理へリダイレクト リダイレクト: "/～/delete/check"
処理2	(削除確認画面表示処理) path = "/～/delete/check", method = RequestMethod.GET ・セッションスコープから入力フォーム情報を取得 ・入力フォーム情報をリクエストスコープに設定 ・削除確認画面表示 フォワード :“～/delete_check”
処理3	(削除ボタン 押下時処理) path = "/～/delete/complete", method = RequestMethod.POST ・セッションスコープから入力フォーム情報を取得 ・入力フォーム情報を元に DB 登録用エンティティオブジェクトを生成 ・DB 更新実施 ・セッションスコープの入力フォーム情報削除 ・処理内容に応じて、セッションスコープの内容を書き換える - ログインユーザの会員退会の場合、セッションスコープの情報を破棄 - カテゴリ削除の場合、セッションスコープのカテゴリー覧を更新 ・削除完了画面表示処理にリダイレクト リダイレクト :"/～/delete/complete"
処理4	(削除完了画面表示処理) path = "/～/delete/complete", method = RequestMethod.GET ・登録完了画面表示 フォワード :“～/delete_complete”

4 - 7 買い物かごの構造

・買い物かごの処理は、/client/basket/ClientBasketController クラスで行う
・買い物かご情報は、セッションスコープで維持し、注文完了処理、ログアウト処理、退会処理で削除する
・BasketBean クラスを利用し、買い物かご表示内容を生成する

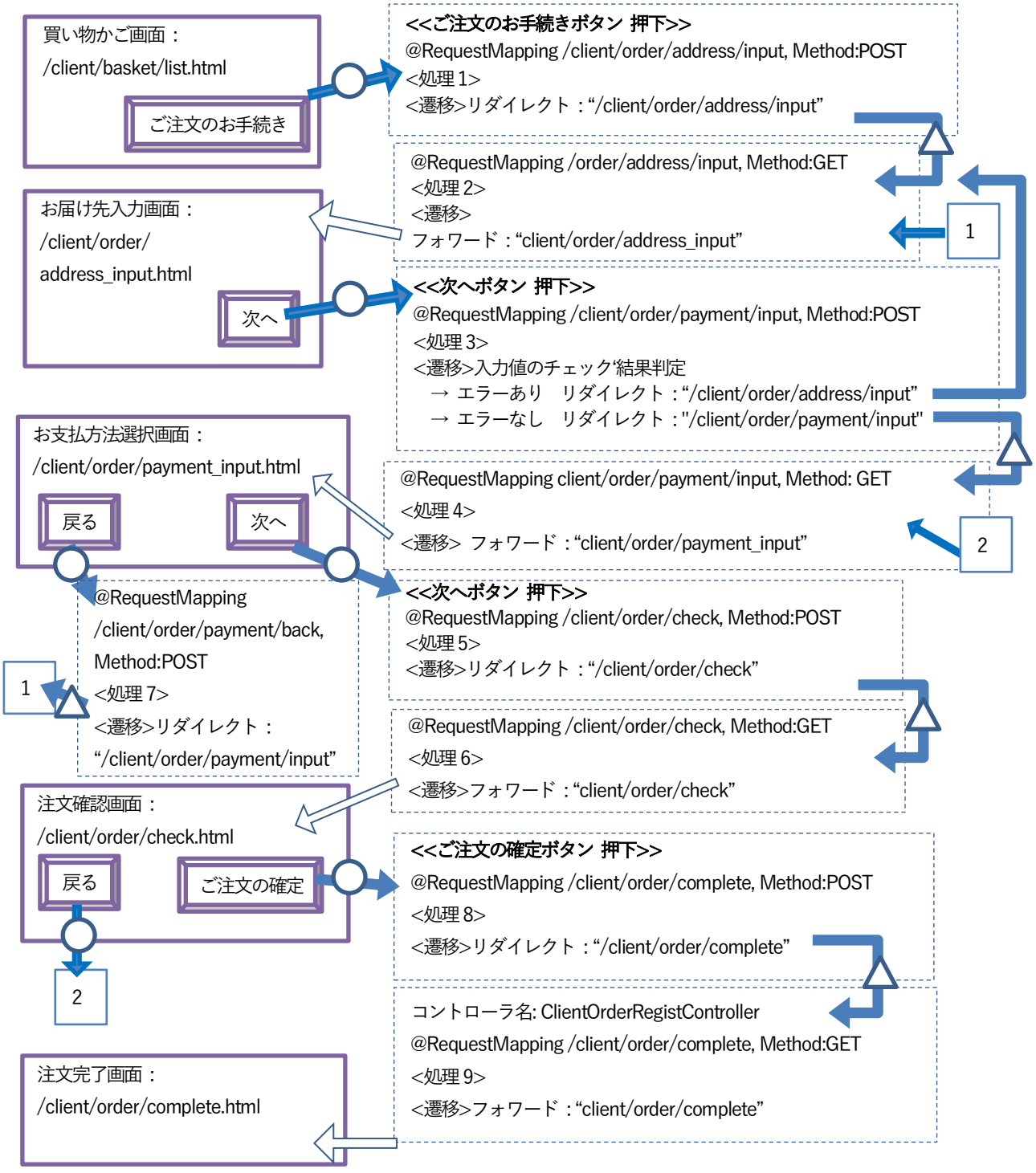
(1)画面遷移とコントローラクラスの関係
凡例　△:GET、○:POST



4 - 8 注文登録処理の構造

・「戻る」ボタン押下時に URL 維持、入力値、入力チェック結果を維持できるようにする
・処理は/client/order/ClientOrderRegistController クラスで実装する

(1)画面遷移とコントローラクラスの関係
凡例　△:GET、○:POST



(2)コントローラクラス処理概要

ClientOrderRegistController 処理

処理	マッピング, 処理内容
処理 1	(ご注文のお手続きボタン 押下時処理) path = "/client/order/address/input", method = RequestMethod.POST ・注文入力フォーム情報初期化 - 注文入力フォーム情報を作成 - セッションスコープからログイン会員情報を取得 - 取得したログイン会員情報のユーザ ID を条件に DB からユーザ情報を取得 - 取得したユーザ情報を注文入力フォーム情報に設定 - 注文入力フォーム情報の支払方法に初期値としてクレジットカードを設定 ・注文入力フォーム情報をセッションスコープに保存 ・届け先入力画面表示処理へリダイレクト - リダイレクト:" /client/order/address/input "
処理 2	(届け先入力画面表示処理) path = "/client/order/address/input", method = RequestMethod.GET ・セッションスコープから注文入力フォーム情報を取得 ・注文入力フォーム情報をリクエストスコープに設定 ・セッションスコープに入力エラー情報がある場合 - 取得したエラー情報をリクエストスコープに設定 - セッションスコープから、エラー情報を削除 ・登録画面表示 - フォワード:"client/order/address_input
処理 3	(届け先入力画面 次へボタン 押下時処理) path = "/client/order/payment/input", method = RequestMethod.POST ・リクエストスコープに画面から入力されたフォーム情報を注文入力フォーム情報として保存 ・BindingResult オブジェクトに入力エラー情報がある場合 - 入力エラー情報をセッションスコープに設定 - 届け先入力画面表示処理にリダイレクト リダイレクト :“/client/order/address/input” ・入力エラーがない場合 - 支払方法選択画面表示処理にリダイレクト リダイレクト :“/client/order/payment/input”

処理	マッピング, 処理内容
処理 4	(支払方法選択画面表示処理) path = "/client/order/payment/input", method = RequestMethod.GET ・セッションスコープから注文入力フォーム情報を取得 ・注文フォーム情報をリクエストスコープに設定 ・支払方法選択画面表示 フォワード : “client/order/payment_input”
処理 5	(支払方法選択画面 次へボタン 押下時処理) path = "/client/order/check", method = RequestMethod.POST ・セッションスコープから注文入力フォーム情報を取得 ・画面から入力された支払方法を取得した注文入力フォーム情報に設定 ・注文入力フォーム情報をセッションスコープに保存 ・注文確認画面表示処理へリダイレクト リダイレクト : “/client/order/check”
処理 6	(注文確認画面表示処理) path = “/client/order/check”, method = RequestMethod.GET ・セッションスコープから注文情報を取得 ・セッションスコープから買い物かご情報を取得 ・注文商品の最新情報を DB から取得し、の在庫チェックをする ・在庫不足、在庫切れ商品がある場合 - 注文警告メッセージをリクエストスコープに保存 - 在庫数にあわせて、買い物かご情報を更新（注文数、在庫数） - 在庫切れの商品は、買い物かごか情報ら削除 ・在庫状況を反映した買い物かご情報をセッションに保存 ・買い物かご情報から、商品ごとの金額小計と全額を算出し、注文入力フォーム情報に設定 ・注文入力フォーム情報をリクエストスコープに設定 ・注文確認画面表示 フォワード: "client/order/check"
処理 7	(注文確認画面で、戻るボタン押下処理) path = “/client/order/payment/back”, method = RequestMethod.POST ・支払い方法選択画面 表示処理へリダイレクト リダイレクト: "/client/order/payment/input"

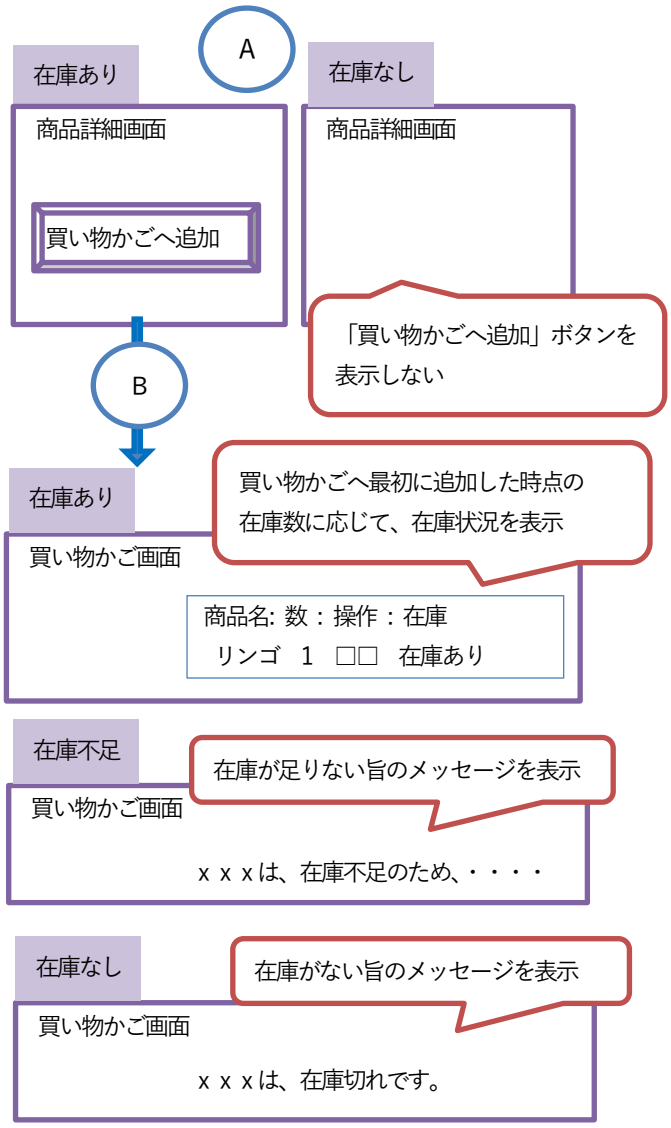
処理	マッピング, 処理内容
処理 8	(ご注文の確定ボタン 押下時処理) path = "/client/order/complete", method = RequestMethod.POST ・セッションスコープから注文情報を取得 ・セッションスコープから買い物かご情報を取得 ・注文商品の在庫チェックをする ・在庫切れ商品がある場合 -注文確認画面表示処理へリダイレクト リダイレクト:" /client/order/check" ・注文情報情報を元に DB 登録用エンティティオブジェクトを生成 ・注文テーブルおよび注文商品テーブルの DB 登録実施 ・セッションスコープの注文入力フォーム情報と買い物かご情報を削除 ・注文完了画面表示処理にリダイレクト リダイレクト : "/client/order/complete"
処理 9	(注文完了画面表示処理) path = "/client/order/complete", method = RequestMethod.GET ・注文完了画面表示 フォワード : "client/order/complete"

4 - 9 在庫数の扱い

在庫数を扱うタイミングを明確にしておくことで、システム内で不整合が発生しにくい構造にする。

(1) 商品詳細、買い物かご機能と在庫数の扱い

<画面遷移>



<在庫数を扱う処理>

A

商品詳細画面表示処理で、在庫数を取得
テンプレートで在庫数により表示内容を変更
在庫あり→買い物かごボタンを表示する
在庫なし→買い物かごボタンを表示しない

B

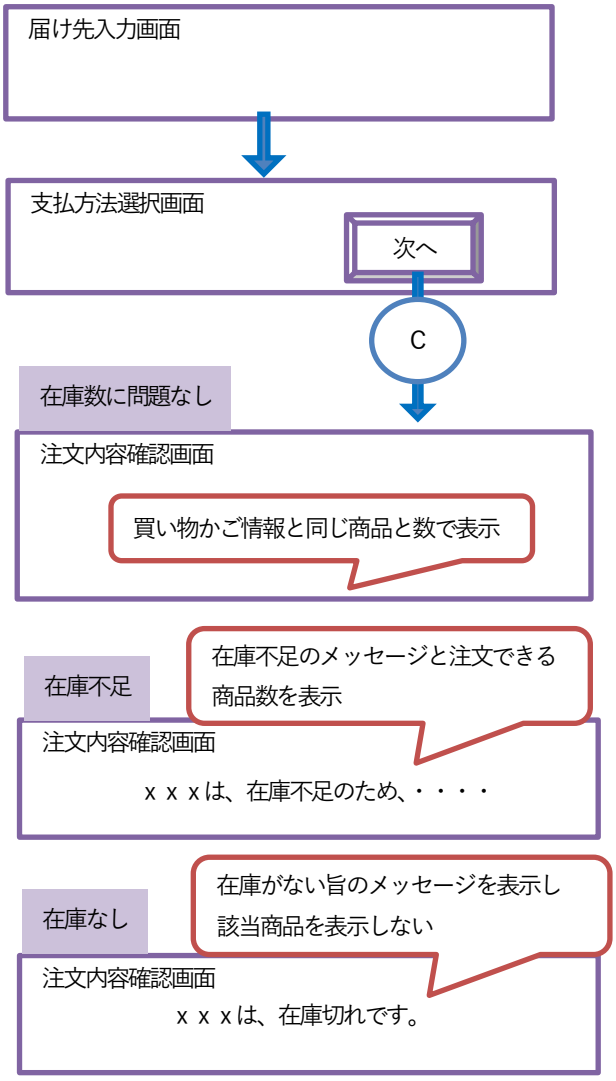
買い物かご表示処理で、在庫数を確認
買い物かごに同じ商品があるかを確認
同じ商品あり→
　最初に買い物かごに追加した時点の在庫数が、
　買い物かご内の数以下である
　→在庫不足のメッセージを表示
　最初に買い物かごに追加した時点の在庫数が、
　買い物かご内の数より多い
　→数を増やす
同じ商品なし→
在庫あり→買い物かごへ追加
在庫なし→在庫切れのメッセージを表示

考慮点:
商品詳細表示時点や買い物かごに初めて追加した時点で、在庫ありのものが、買い物かごへボタン押下時に、在庫なし、在庫不足になる可能性を考慮

(2) 注文内容確認画面と在庫数の扱い

注文対象の商品と商品数を確定するために、在庫数を確認する。

<画面遷移>



<在庫数を扱う処理>

C

注文内容確認処理で、買い物かごに入っている商品の在庫数と注文数を比較。

注文数より、在庫数が少ない商品(在庫不足)

→数不足のメッセージを表示し、
在庫数分を注文数として注文商品に表示

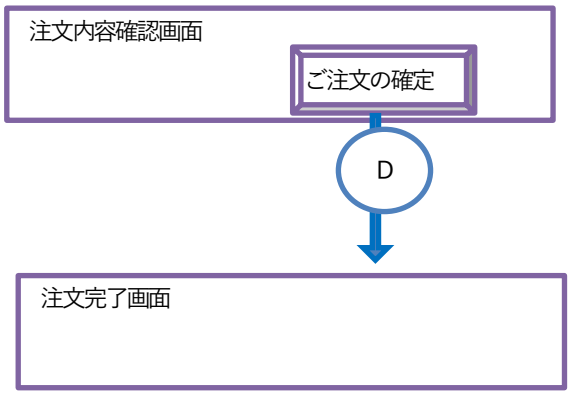
在庫なしの商品

→在庫がないメッセージを表示し、
注文商品には表示しない

考慮点:
買い物かご追加時点から注文内容確認までの間に
在庫数が変化していることを想定

(3) 注文情報の反映と在庫数の扱い

<画面遷移>



<在庫数を扱う処理>

D

注文内容確認画面で表示した注文数を発注数として扱う。
そのため、「ご注文の確定」ボタン押下後は
下記の処理を行う。

- 注文情報のデータベースへの登録
- 注文商品情報のデータベースへの登録
- 商品情報の在庫数を変更(在庫数を減らす)

考慮点:
注文内容録確認画面を表示した時点から「ご注文の確定」ボタンを押されるまでに在庫数が変化することを想定し、
DB 登録の直前処理として、注文商品の在庫数を DB へ問合せをする。
注文数が在庫数に合わせて変更となる場合は、
注文内容確認画面を再表示するとともに、その旨を警告メッセージとして表示する。
なお、全注文商品が在庫切れとなる場合は、注文を中止するメッセージを表示する。

4-10エラー処理の構造

例外エラー発生時は、Spring フレームワークにより、予め準備してある error 画面テンプレート(error.html)画面に遷移する。
ロジック内で error とする場合は、エラー処理へのリダイレクトを記述する。

・リダイレクト："/syserror"

5 技術的な制約・既知の問題点

(1) 商品登録・商品変更機能について

・商品登録、商品変更の入力画面で商品画像ファイルを選択し、確認画面へ遷移

・確認画面で「戻る」ボタンを押し、入力画面に遷移する場合、最初に選択した画像ファイル名が維持できない

(2) 商品一覧画面の検索条件・表示順条件について

・商品一覧画面で、カテゴリ別・表示順変更を行った後、商品詳細画面に遷移、その後「戻る」ボタンで、
商品一覧画面に戻った場合、条件が維持できない

(3) 入力チェックについて

・必須入力の入力欄に、全角空白文字のみを入力しても、未入力と判断されない

・数値のみを入力する欄に、数値のみを 10 桁以上入力した場合、数値の範囲指定のエラーではなく、
「数値で入力してください」というメッセージが表示される。

(4) 画像ファイルの反映について

・Eclipse 環境でサーバ(SpringBoot アプリケーション(Tomcat))を実行し、ファイルをアップロードした後、サーバを再起動した際に画像が存在しないと判断されてしまう。Eclipse でプロジェクトをリフレッシュすることで、正常に反映される