

JavaWeb システム修了問題

●仕様書を基にしたコーディング

プログラマがシステムの開発をする際には、開発するシステムの詳細な仕組み、内容が記述された「仕様書」という書類に基づいてコーディングしていきます。

プログラマは、ただプログラムを書けばよいのではなく、仕様書に書かれている内容を理解しシステムの仕組みを把握し、それに従い正確に指示を満たすプログラムを書いていくことが求められます。

今回は開発現場を想定し、仕様書を解読してのコーディングに挑戦しましょう。

●前提条件

ここからショッピングサイトのシステムを構築していきます。

開発現場では、チームで分担しコーディングを行います。今回は開発チームの一員として一部すでに構築されているシステムの開発に途中から携わることを前提とします。

●開発環境・運用環境

今回作成するショッピングサイトシステムの、開発時、運用時に想定されている環境です。

	開発環境	運用環境
JavaSE	Java11	Java11
サーブレットコンテナ	Tomcat9	Tomcat9
Web サーバー	-	Apache2.4
データベース	MySQL8.0	MySQL8.0
OS	Windows10	Linux(CentOS6.4)

運用するための環境とプログラマがプログラムをコーディングしていく開発環境は、異なる場合があります。多くの場合、開発は WindowsOS 上で行い、運用時には LinuxOS のサーバー上にプログラムを乗せかえるという作業を行います。

したがって、Java のプログラマでも、LinuxOS を操作するためのコマンドの知識が要求される場面が多くあることを覚えておきましょう。

LinuxOS の講座については、インストラクターにお尋ねください。

【目安時間：180分】

●ショッピングサイトシステム 画面遷移と機能

商品検索画面

【機能】

- ・ 商品検索機能（分類検索、キーワード検索）
- ・ 商品を買い物カゴに登録する機能
- ・ 買い物カゴの商品数を確認する機能
- ・ 買い物カゴ画面に遷移する機能



買い物カゴ画面

【機能】

- ・ 買い物カゴの中の商品一覧を表示する機能
- ・ 買い物カゴの商品を削除する機能
- ・ 商品注文画面に遷移する機能
- ・ 商品検索画面に遷移する機能



買い物カゴに商品が無いのに注文するボタンを押した場合

【機能】

- ・ エラーメッセージを表示する機能
- ・ 商品検索画面に遷移する機能



エラー

現在買い物カゴには何も入っていないので、注文することはできません。
TOP画面へ戻るで戻ってください。

[TOP画面へ戻る](#)

商品注文画面

【機能】

- ・ 買い物カゴの中身一覧を表示する機能
- ・ 購入者の個人情報を入力して注文完了画面に送信する機能
- ・ 個人情報入力欄が1箇所でも未入力の場合、アラート画面を出す機能
- ・ 注文完了画面に遷移する機能
- ・ 買い物カゴ画面に遷移する機能



Latte station 個人情報入力画面

※ 必須項目は赤字で表示します。

お名前 山田太郎

フリガナ ヤマダ タロウ

郵便番号 〒113-0032

住所 東京都中央区銀座1-1-1 1F 101号室

電話番号 (03) 1234-5678

メールアドレス hogehoge@example.jp

※ 上記の項目で入力してください。入力漏れはできません。

[商品検索画面に戻る](#)

未入力

Latte station 個人情報入力画面

※ 必須項目は赤字で表示します。

お名前 山田太郎

フリガナ ヤマダ タロウ

郵便番号 〒113-0032

住所 東京都中央区銀座1-1-1 1F 101号室

電話番号 (03) 1234-5678

メールアドレス hogehoge@example.jp

※ 上記の項目で入力してください。入力漏れはできません。

[商品検索画面に戻る](#)

【目安時間：180分】

注文完了画面

【機能】

- ・ 注文確定機能(データベースに注文内容、個人情報を追加する)
- ・ 商品検索画面に遷移する機能



■ 注文完了

ご注文ありがとうございます。
24時間以内に当サイト管理者よりメールがまいりますので、よろしくお願い致します。
在庫の関係で商品が遅れる場合がございますので、あらかじめご了承ください。

[TOP画面へ戻る](#)

エラー画面

【機能】

- ・ エラーメッセージを表示する機能
- ・ 商品検索画面に遷移する機能



■ エラー

エラーが発生しました。管理者に問い合わせてください。

[TOP画面へ戻る](#)

●ショッピングサイトシステム詳細

ここからは、ショッピングサイトシステムの詳細な動きを確認します。

商品検索画面

この画面では買い物をするために商品の検索をします。

分類には、書籍、CD、GAME の 3 種類があります。キーワードの欄にはその商品の名称もしくはメーカーなどを入れて検索ボタンを押すと商品が表示されます。




【目安時間：180 分】

分類のプルダウンメニューから、「書籍」を選択し、キーワードを入れずに [検索] ボタンを押すと書籍カテゴリーの商品が全て表示されます。

商品	商品名 著者 価格	
	ホリー・ポッター ケン・ジャイシー 580	🛒 カートに追加
	AGE of vase ダンレッド 410	🛒 カートに追加
	龍玉60年の歴史 島山明 600	🛒 カートに追加
	メサニヨキ さくらむむこ 1050	🛒 カートに追加

分類 書籍 ▼ キーワード × 検索

分類のプルダウンメニューから、「書籍」を選択し、さらにキーワードに「ホリー」と入力して検索ボタンを押すと、商品名か著者に「ホリー」という文字列が含まれている商品が検索結果として表示されます。

商品	商品名 著者 価格	
	ホリー・ポッター ケン・ジャイシー 580	🛒 カートに追加

分類 書籍 ▼ キーワード × 検索


キーワードに、取り扱っていない商品名や著者名などを入力すると、「該当する商品は存在しません。」と表示されます。

商品	商品名 著者 価格	
----	-----------	--

該当する商品はありません。

【目安時間：180 分】

商品を買い物カゴに登録する機能を見ていきます。

商品	商品名 著者 価格	
	ホリー・ポッター ケン・ジャイシー 580	🛒 カートに追加

表示されている商品の右にある [カートに追加] ボタンを押すと現在選択されている商品が一つ増えます。[カートに追加] ボタンを押すたびに増えていきます。

現在選択されている商品

選択商品 1個

[🛒 カートの中身を見る](#)

この状態で [カートの中身を見る] ボタンを押すと、買い物カゴ画面に遷移します。

買い物カゴ画面

ここでは、選んだ商品を確認して、商品注文画面、もしくは商品検索画面に遷移します。

■ 買い物カゴの中身

↓↓現在買い物カゴには以下の商品が入っています。↓↓

タイトル	メーカー等	価格	
ホリー・ポッター	ケン・ジャイシー	580	取り消し

↓↓上記の内容で注文画面へ進む場合はこちら↓↓

 [上記内容で注文する](#)

↓↓まだ買い物を続けたいので検索画面に戻る場合はこちら↓↓

[検索画面に戻る](#)

表示されている商品の右にある[取り消し] ボタンを押すと、このように先程まで入っていた商品は買い物カゴの中から削除されます。

■ 買い物カゴの中身

↓↓現在買い物カゴには以下の商品が入っています。↓↓

タイトル	メーカー等	価格	
------	-------	----	--

↓↓上記の内容で注文画面へ進む場合はこちら↓↓

 [上記内容で注文する](#)

↓↓まだ買い物を続けたいので検索画面に戻る場合はこちら↓↓

[検索画面に戻る](#)

買い物カゴに商品が無い状態で、[上記内容で注文する] ボタンを押すと、エラー画面に遷移します。

■ エラー

現在買い物カゴには何も入っていないので、注文することはできません。
TOP画面へ戻るで戻ってください。

[TOP画面へ戻る](#)

商品がカートの中に入っている状態で [上記内容で注文する] ボタンを押すと、注文画面に遷移します。

【目安時間：180 分】

商品注文画面

送付先登録画面

買い物の内容は以下になります

タイトル	メーカー等	価格
ホリー・ポッター	ケン・ジャイシー	580

合計金額は税込みで580円になります。

下のフォームにお客様のお名前、ご住所、電話番号、メールアドレスを入力して、よろしければ「注文する」ボタンを押してください。

項目は必ず入力してください。機種依存文字は入力できません。

お名前	<input type="text"/> (例) 山田太郎
フリガナ	<input type="text"/> (例) ヤマダタロウ
郵便番号	<input type="text"/> (郵便番号検索は 検索) (例) 153-0052 ※ 国外の方は000-0000 と入力してください。
住所	<input type="text"/> (例) 東京都世田谷区北沢1-1-1KENコーポ203号室
電話番号	<input type="text"/> (例) 03-0000-0000
メールアドレス	<input type="text"/> (例) hogehoge@kenschool.jp
↓↓上記の内容で問題なければ、注文を確定いたします↓↓	

このように買い物カゴの中身と合計金額が上位に表示され、さらに下位に住所などを入力する画面が表示されます。

ここで名前や住所などを入力しますが、入力していない欄がある状態で「注文する」ボタンを押した場合は、このように「入力されていない項目があります。」というアラート画面が JavaScript で表示されます。

※ブラウザにより表示される画面は変わります



The screenshot shows a web browser window with a title bar indicating 'localhost:8080'. An alert dialog box is displayed in the foreground with the text 'localhost:8080 の内容' and '入力されていない項目があります。' (There are items that have not been entered). Below the alert, the page content is partially visible. It includes a section titled '送付元登録画面' (Sender Registration Screen). Below this, it says '買い物の内容は以下になります' (The purchase content is as follows). A table lists the items in the cart:

タイトル	メーカー等	価格
ホリー・ポッター	ケン・ジャイシー	580

Below the table, it states '合計金額は税込みで580円になります。' (The total amount including tax is 580 yen). It then prompts the user to enter their name, address, phone number, and email address to proceed with the order. A note says '項目は必ず入力してください。機種依存文字は入力できません。' (Please enter all items. Device-dependent characters cannot be entered). At the bottom, there is a form with a label 'お名前' (Name) and a text input field containing '(例) 山田太郎' (Example: Yamada Taro).

すべてを入力し「注文する」ボタンを押した場合は、注文完了画面に遷移します。

【目安時間：180 分】

注文完了画面

このように注文完了となります。

■ 注文完了

ご注文ありがとうございます。

24時間以内に当サイト管理者よりメールがまいりますので、よろしくお願い致します。

在庫の関係で商品が遅れる場合がございますので、あらかじめご了承ください。

[TOP画面へ戻る](#)

注文情報のデータベースへの反映

なお、この際にユーザの情報がデータベースの user テーブルに、注文情報が orderitem テーブルに追加されます。

DBViewer で確認すると、

【user テーブル】

	user_id	user_name	user_name_kana	post	address	phone	mail
1	1001	野村朱里	ノムラアカリ	932-3333	東京都世田谷区北沢1-1-1KENコーポ203	090-0000-0000	kenken1@kenschool.jp
2	1002	佐藤賢一	サトウケンイチ	932-3334	千葉県船橋市1-1-1KENコーポ204	090-0000-0000	kenken2@kenschool.jp
3	1003	田中秀雄	タナカヒデオ	932-3335	神奈川県横浜市1-1-1KENコーポ205	090-0000-0000	kenken3@kenschool.jp
4	1004	松本昭徳	マツモトアキノリ	932-3336	神奈川県秦野市1-1-1KENコーポ204	090-0000-0000	kenken4@kenschool.jp
5	1005	山田太郎	ヤマダタロウ	153-0052	東京都世田谷区北沢1-1-1KENコーポ203号室	03-0000-0000	hogehoge@kenschool.jp

【orderitem テーブル】

	order_id	user_id	item_id
1	1	1005	101

先程の情報が書き込まれていることがわかります。

こうすることで、管理者が後に商品を配送する際、誰がどの商品を買ったかを確認することができます。

●システム設計

プロジェクト名

ken_shop

URL

"Control.java" ファイルを右クリック → [実行] → [サーバーで実行(1)] → [次へ(N)] → [完了] で起動

http://localhost:8080/ken_shop/kenshop

データベース

データベース名	latte_station	
テーブル名	item	商品情報を格納するテーブルです
	user	商品を注文したユーザ情報を格納するテーブルです
	orderitem	注文された履歴を格納するテーブルです

※ ワークスペースを今までとは変えて最終課題を行う場合は、必ずサーバーに mysql-connector-java-5.1.47-bin.jar ファイルの設定を行ってください

プログラム構成

項目		値
1	動的 Web プロジェクト名	ken_shop
2	パッケージ名	ken
	サーブレットクラス名	Control
	URL マッピング	/kenshop
	メソッド・スタブ	<input checked="" type="checkbox"/> 継承された抽象メソッド(H) <input checked="" type="checkbox"/> doGet()(G) <input checked="" type="checkbox"/> doPost()(P)
3	パッケージ名	ken.act
	クラス名	Action
	修飾子	abstract
4	パッケージ名	ken.act
	クラス名	AddAction
	継承元	Action
	メソッド・スタブ	<input checked="" type="checkbox"/> 継承された抽象メソッド(H)
5	パッケージ名	ken.act

【目安時間：180 分】

	クラス名	CheckAction
	継承元	Action
	メソッド・スタブ	<input checked="" type="checkbox"/> 継承された抽象メソッド(H)
6	パッケージ名	ken.act
	クラス名	For_OrderAction
	継承元	Action
	メソッド・スタブ	<input checked="" type="checkbox"/> 継承された抽象メソッド(H)
7	パッケージ名	ken.act
	クラス名	OrderAction
	継承元	Action
	メソッド・スタブ	<input checked="" type="checkbox"/> 継承された抽象メソッド(H)
8	パッケージ名	ken.act
	クラス名	RemoveAction
	継承元	Action
	メソッド・スタブ	<input checked="" type="checkbox"/> 継承された抽象メソッド(H)
9	パッケージ名	ken.act
	クラス名	SearchAction
	継承元	Action
	メソッド・スタブ	<input checked="" type="checkbox"/> 継承された抽象メソッド(H)
10	パッケージ名	ken.act
	クラス名	TopAction
	継承元	Action
	メソッド・スタブ	<input checked="" type="checkbox"/> 継承された抽象メソッド(H)
11	パッケージ名	ken.bean
	クラス名	Item
	インタフェース	java.io.Serializable
	メソッド・スタブ	<input type="checkbox"/> 継承された抽象メソッド(H)
12	パッケージ名	ken.bean
	クラス名	User
	インタフェース	java.io.Serializable
	メソッド・スタブ	<input type="checkbox"/> 継承された抽象メソッド(H)
13	パッケージ名	ken.dao
	クラス名	OrderDAO
14	パッケージ名	ken.dao
	クラス名	SearchDAO
15	JSP ファイル名	cart.jsp
16	JSP ファイル名	error.jsp

17	JSP ファイル名	finish.jsp
18	JSP ファイル名	irregular_error.jsp
19	JSP ファイル名	order.jsp
20	JSP ファイル名	top.jsp
21	css	WebContent/css
22	img	WebContent/img
23	プロパティファイル	WebContent/WEB-INF/action.properties

ファイル一覧

JAVA クラス	
■ken パッケージ	今回作成するクラス全てが所属するパッケージです。
Control.java	コントローラの役割をするサーブレットです。 今回は、リクエストは全て Control サーブレットに送信され、そこでリクエストごとの処理に分岐されます。
■ken.act パッケージ	リクエストごとの機能を持つクラスが所属するパッケージです。
Action.java	リクエストごとの機能を持つクラスの親クラスです。具体的な処理を記述するクラスは、全てこのクラスを継承し、各機能は execute() メソッドをオーバーライドして記述します。
TopAction.java	検索画面に遷移する機能を持つクラスです。
SearchAction.java	検索機能を持つクラスです。
AddAction.java	買い物カゴに商品を追加する機能を持つクラスです。
CheckAction.java	買い物カゴの中身を見るページに遷移する機能を持つクラスです。
RemoveAction.java	買い物カゴから商品を削除する機能を持つクラスです。
For_OrderAction.java	注文ページに遷移する機能を持つクラスです。
OrderAction.java	注文機能を持つクラスです。
■ken.bean パッケージ	Bean クラスが所属するパッケージです。
User.java	ユーザ情報を保持する Bean です。
Item.java	商品情報を保持する Bean です。
■ken.dao パッケージ	DB 接続処理をするクラスが所属するパッケージです。
OrderDAO.java	発注に使用する DB 接続クラスです。
SearchDAO.java	検索に使用する DB 接続クラスです。
JSP ファイル	
top.jsp	トップの検索画面です。
cart.jsp	買い物カゴの中身を表示する画面です。

【目安時間 : 180 分】

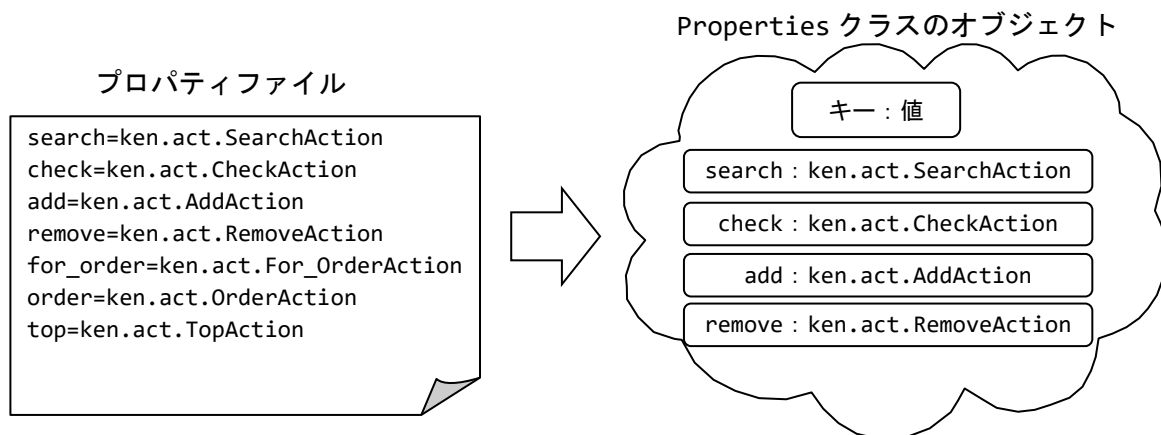
order.jsp	商品注文画面です。
finish.jsp	注文完了画面です。
error.jsp	買い物カゴが空の時に、注文画面へ遷移した場合のエラー画面です。
irregular_error.jsp	その他のエラー画面です。
その他設定ファイル	
action.properties	リクエストと、機能ごとのクラスを関連付けするためのプロパティファイルです。

※ プロパティファイルについて

今回、サーブレットでのコントロールのために、プロパティファイルという形式のテキストファイルを利用しています。

プロパティファイルとは「キー=値」という形式でデータを持つ Java のアプリケーションから利用されるテキストファイルのことです。拡張子は「.properties」にします。

このプロパティファイルの内容は、Java プログラムから読み込み、java.util.Properties というクラスを用いて、「キー : 値」のマップのような形式で利用することが可能です。



例)

プロパティファイル 「〇〇.properties」

```
search=ken.act.SearchAction
check=ken.act.CheckAction
add=ken.act.AddAction
remove=ken.act.RemoveAction
for_order=ken.act.For_OrderAction
order=ken.act.OrderAction
top=ken.act.TopAction
```

Java プログラム

```
//プロパティファイルまでのストリームを構築
FileInputStream stream = new FileInputStream("プロパティファイルまでのパス");

//Properties クラスの load()メソッドを用いて、プロパティファイルの内容を読み込み。
Properties props = new Properties();
props.load(stream);

//キーを指定して、値に設定された文字列を取得
String value =props.getProperty("キー");
```

今回の Web アプリケーションは、各リクエストに、どのボタンやリンクがクリックされたかを判別するためのリクエストパラメータが付加されるようになっています。

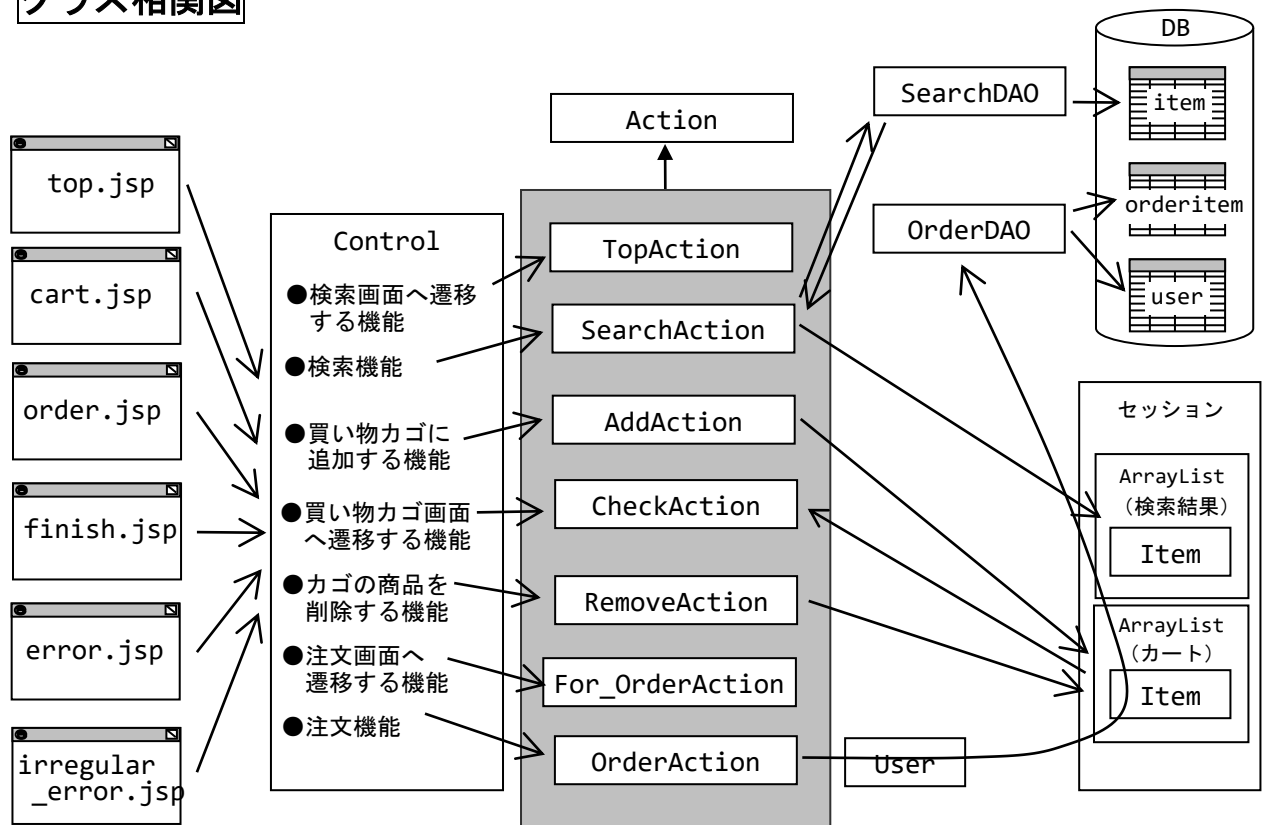
プロパティファイル「action.properties」には、このリクエストパラメータをキーとして、そのときの処理内容が定義されているクラス名が値に設定されています。

Control サーブレット内では、リクエストパラメータにしたがって、そのリクエストの際に処理すべきクラス名をプロパティファイル「action.properties」から読み取り、インスタンス化して実行しています。

ファイル配置

- ▼ ken_shop
 - > デプロイメント記述子: ken_shop
 - > JAX-WS Web サービス
 - ▼ Java リソース
 - ▼ src
 - ▼ ken
 - > Control.java (作成済み)
 - ▼ ken.act
 - > Action.java (作成済み)
 - > AddAction.java (新規作成)
 - > CheckAction.java (新規作成)
 - > For_OrderAction.java (作成済み)
 - > OrderAction.java (作成済み)
 - > RemoveAction.java (新規作成)
 - > SearchAction.java (新規作成)
 - > TopAction.java (新規作成)
 - ▼ ken.bean
 - > Item.java (新規作成)
 - > User.java (作成済み)
 - ▼ ken.dao
 - > OrderDAO.java (作成済み)
 - > SearchDAO.java (新規作成)
 - > ライブラリー
 - > 参照ライブラリー
 - > build
 - ▼ WebContent
 - > css (各種スタイルシートファイル)
 - > img (各種画像ファイル)
 - > META-INF
 - ▼ WEB-INF
 - ▼ lib
 - > jstl-1.2.jar
 - > mysql-connector-java-5.1.47-bin.jar
 - action.properties (作成済み)
 - cart.jsp (未完成)
 - error.jsp (作成済み)
 - finish.jsp (作成済み)
 - irregular_error.jsp (作成済み)
 - order.jsp (未完成)
 - top.jsp (未完成)

クラス相関図



今回の Web アプリケーションは、リクエストは全て、Control サーブレットに送信されます。

また、リクエストごとの具体的な機能は、Action クラスを継承した〇〇Action というクラスに全て記述されています。

この様に、次にどこに処理を移すかを 1 箇所で行き先を分岐して処理をまとめる仕組みを今回の仕様では採用しています。こういった設計は、JavaEE のデザインパターンの 1 つで「Front Controller」とよばれています。

●開発手順

配布物のインポート

動的 Web プロジェクトを「ken_shop」という名前で作成し、配布された ken_shop.zip を解凍し、インポートします。作成済みファイルと、未完成ファイルがインポートされていることを確認してください。

データベースの作成

配布資料の「latte_station.txt」に書かれている SQL 文を発行し、データベースの作成、テーブルの作成を行ってください。

コーディング

- ① 次のファイルを作成すると商品検索画面の機能を確認することができます

Item.java
SearchDAO.java
TopAction.java
SearchAction.java
AddAction.java
top.jsp

- ② 次のファイルを作成すると、一通りの機能が出来上がります

CheckAction.java
RemoveAction.java
cart.jsp
order.jsp

テスト

プログラマの仕事は、プログラムを書き終えたら終わりではありません。

作成したプログラムがどのような操作をしてもエラーになることなく、目的どおりの処理を確実に行うことを確認するまでがプログラマの仕事です。

その、プログラムが確実に動作するかを確認する作業を「テスト」といいます。通常テストは、コーディング後、またはコーディングと同時進行に行われ、開発の中で製品の品質を決める非常に重要な工程でもあります。

テストを専門で担当する人のことを「テスター」と呼ぶ場合もあります。

テストのなかでプログラマが自身で担当した箇所に関して、テストをすることを「単体テスト」といい、他のプログラマが作成した箇所と合わせて動作を確認することを「結合テスト」といいます。

今回は、コーディングの①を作成した段階と、②を作成した段階で「単体テスト」を行い、ご自身で作成された機能の動作確認を行ってください。この指示書の、画面遷移や、機能の詳細説明と照らし合わせて、目的どおりの処理を行うかの確認をしましょう。

また、全て作成したところで、「結合テスト」として、全体の動きを確認してください。

インストラクターによる確認

コーディングの①の単体テストを終えた段階、②の単体テストを終えた段階、全体の結合テストを終えた段階では、それぞれインストラクターに必ず声をかけ、動作の確認を行ってください。

その際に、画面上での動作確認だけではなく、プログラムの構成、処理の流れが理解できているかの確認も行ってください。