스프링 기말 프로젝트(개발자 매뉴얼)

-

날짜	2023년 06월 13일	전공	소프트웨어학과
과목	스프링 프레임워크	담당교수	신 우 창 교수님
팀 이름 : 김 홍 국			
학번	2018301017	이름	김 주 홍
학번	2018301046	이름	이 동 국

스프링 기말 프로젝트 [POS]

작성 일시 2023.06.12 **작성자** 김주홍, 이동국

팀명	김홍국		
팀원	이동국	팀원	김주홍
학번	20183010	학번	201830107
프로젝트명	편의점 POS(Point of Sale)		
프로젝트 목적	- 편의점 POS 웹 시스템 개발 - spring mvc에 대한 이해 - 개인 코딩 능력 확인 및 향상 - 팀 프로젝트로 인한 협동심 향상		
프로젝트 소개	현재 사용되고 있는 편의점 POS기를 벤치마킹하여 개발 POS기를 사용하는 사용자(매니저, 직원)로 회원가입 후 로그인하여 사용 가능하며 매니저로 로그인 할 경우 판매할 제품별로 코드와 가격을 지정하고 날짜별 입고수량등을 관리하고 직원인 경우 고객이 가져온 제품의 코드를 작성하면 그 제품의 코드와 가격 수량을 계산하여 판매한다. 마지막으로 하루/일주일/한달 간격으로 판매량과 매출액, 그리고 최다 판매제품을 통계자료로 확인할 수 있다.		

- API 설계

	API	Method	URL	요청	응답
회원 관련 API	회원 등록	POST	/members	RegisterRequest 객체 (회원 정보)	HTTP 상태 코드 201 (Created)
	회원 가입	POST	/register	{ “id” : 사용자 아이디, “password” : 사용자 비밀번호, “name” : 사용자 이름, “grade”: 등급}	
	회원 로그인	POST	/login	{ "id": "사용자 ID", “password": "비밀번호" }	{”name”: 사용자 이름 “wallet” : 지갑 잔액 “sale” : 판매량 }
상품 관련 API	상품 조회	POST	/home	{“productCode” : 상품 코드}	{”wallet” : 지갑 잔액, product(List), “sale” : 판매량, “name” : 사용자 이름}
	주문 취소	POST	/cancel	{”cancelCode” : 취소할 상품 이름}	
결제 관련 API	결제	POST	/payment	{ "saleAmount": 결제금액 }	

관리자 기능 관련 API	관리자 로그인	POST	/admin/login	{ "password": "관리자비밀번호" }	HTTP 상태 코드 200 (OK)와 관리자 로그인 결과
	상품 조회	GET	/admin		
	상품 추가	POST	/admin/product/add	{productCode : 상품 코드, productName : 상품 이름, productQuantity : 상품 수량, productPrice : 상품 가격 }	topSellingProducts (List): 일일 최다 판매 목록, weekTopSellingProducts (List): 주간 최다 판매 목록, monthTopSellingProducts (List): 월간 최다 판매 목록, products (List): 상품 목록
	상품 수정	POST	수량증가 : /admin/product/increase-quantity, 수량 감소 : /admin/product/decrease-quantity	{"productCode" : 상품 코드}	
	상품 삭제	DELETE	/admin/product/delete	{"productCode" : 상품 코드}	

- 요구사항 명세

1. 회원 가입 페이지

- Account 회원가입 버튼을 클릭하기
- 아이디, 비밀번호, 비밀번호 확인, 이름, 등급(관리자or직원) 정보들을 입력하기
- 비밀번호 확인은 비밀번호와 정확하게 일치하기
- 사용자는 회원으로 등록할 수 있어야 함
- 회원가입 버튼을 누르고 에러메세지가 발생하지 않는다면 로그인 페이지로 이동시키기

2. 로그인 페이지

- 로그인, 회원가입 버튼을 만들기
- 아이디, 비밀번호 입력란 만들기
- 관리자용 로그인과 직원용 로그인을 따로 만들기
- 로그인 버튼을 누른 경우 아이디과 비밀번호가 데이터베이스에 등록되어 있는지 확인한 뒤, 하나라도 맞지 않는 정보가 있다면 "Please check your ID and password."라는 메세지를 프론트엔드에서 보여주기
- 로그인에 실패한 경우에는 에러 메시지와 함께 다시 로그인 페이지를 표시
- 로그인 버튼을 눌러서 에러 메세지가 발생하지 않는다면 Home 페이지로 이동

3. 상품 구매 및 결제

- 사용자는 편의점 재고가 있는 상품들에서 선택한 상품을 구매할 수 있어야 함
- 구매한 상품은 상품 목록에서 제거되어야 하고, 지갑 잔액에 구매한 만큼의 상품 가격이 추가되어야 함
- 구매한 상품은 판매 기록에 추가되어야 함
- 판매 기록에는 상품 코드, 상품명, 수량, 가격, 판매 일시 등이 포함되어야 함

4. 홈 페이지

- 사용자가 상품 코드를 입력할 수 있는 입력 필드가 제공되어야 함
- 상품 코드를 입력하고 Register 버튼을 클릭하면 해당 상품의 정보를 화면에 표시
- 상품 정보는 상품 목록에서 조회하여 가져와야 함
- 상품 정보에는 상품코드, 상품명, 가격, 개수이 포함
- 사용자의 지갑 잔액과 현재까지의 총 매출액도 화면에 표시
- 상품 코드를 입력하지 않은 상태에서 Register 버튼을 클릭하면 현재까지의 상품 목록이 화면에 표시

5. 관리자 페이지

- 관리자는 비밀번호를 입력하여 관리자 페이지에 접속할 수 있어야 함
- 비밀번호가 올바르면 관리자 페이지로 이동해야 하고, 그렇지 않으면 에러 메시지와 함께 다시 비밀번호 입력 페이지를 표시해야 함
- 관리자 페이지에는 최근 일, 주, 월간의 매출 순위 상위 상품 목록이 표시되어야 합니다.
- 관리자는 재고 관리를 위해 상품을 추가하거나 삭제할 수 있어야 함
- 상품 추가 및 삭제 후에는 상품 목록이 업데이트되어야 함
- 원하는 상품 추가 시 상품코드, 상품명, 수량, 가격을 명시해야 함

- 주요 기능

1. 회원가입

register2() 메서드 : ID, PASSWORD, NAME, GRADE를 입력해 회원가입

회원 등록을 처리하는 메서드, @RequestParam 어노테이션을 사용하여 HTTP 요청 파라미터를 매핑해 사용자가 입력한 아이디, 비밀번호, 이름, 등급을 파라미터로 받아서 회원 정보를 생성하고, memberDao.InsertMember()를 호출하여 회원 정보를 데이터베이스에 저장

```
@PostMapping(value = "/register")
public String register2(@RequestParam("id") String id, @RequestParam("password") String password,
        @RequestParam("name") String name, @RequestParam("grade") String grade, Model model) {
    Object[] params = {id,password,name,grade};
    memberDao.InsertMember(params);
    return "Login";
}
```

register() 메서드

@GetMapping 어노테이션을 사용하여 GET 요청에 대한 매핑을 설정해 회원 등록 페이지를 보여주는 메서드, 해당 페이지로 이동하면 회원 정보를 입력할 수 있는 폼이 표시

```
@GetMapping(value = "/register")
public String register(Model model) {
    return "register";
}
```

회원가입시 ID와 PASSWORD 입력시 CONFIRMPASSWORD 값과 같지 않으면 경고문 출력

```
<script type="text/javascript">
    function checkForm() {
        if (!document.newMember.id.value) {
            alert("아이디를 입력하세요.");
            return false;
        }

        if (!document.newMember.password.value) {
            alert("비밀번호를 입력하세요.");
            return false;
        }

        if (document.newMember.password.value != document.newMember.password_confirm.value) {
            alert("비밀번호를 동일하게 입력하세요.");
            return false;
        }
    }

</script>
```

회원가입한 정보를 저장해 주는 메서드와 SQL문

```
public void InsertMember(Object[] params) {
    String sql = "INSERT INTO MEMBER(ID, PASSWORD, NAME, GRADE) VALUES(?,?,?,?)";
    jdbcTemplate.update(sql, params);
}
```

2. 로그인

login()메서드 : ID와 PASSWORD를 입력해 로그인

@RequestParam어노테이션을 사용하여 HTTP 요청 파라미터를 매핑해 사용자의 아이디와 비밀번호를 파라미터로 받아서 로그인을 처리하는데 memberDao.selectAll() 함수를 호출하여 데이터베이스에 저장된 회원 정보를 가져온 뒤, 사용자가 입력한 아이디와 비밀번호와 일치하는 회원을 찾습니다.

일치하는 회원이 존재하는 경우) 해당 회원의 이름을 모델에 추가 -> memberinfo.setName()을 함수 호출 -> MemberInfo객체에 사용자의 이름을 설정 -> "redirect:/Home"을 반환해 홈 페이지로 이동

일치하는 회원이 없을 경우) 에러 메시지를 모델에 추가 -> "Login"을 반환해 로그인 페이지에 에러 메시지를 표시

`<div class='alert alert-danger'>${error}</p>` : 에러 창 나오게 함

```
@PostMapping(value = "Login")
public String login(@RequestParam("id") String id, @RequestParam("password") String password, Model model) {
    List<Member> members = memberDao.selectAll();   //데베 저장되어 있는 member 데이터베이스 가져와서 리스트로 저장
    for (Member member : members) {
        if (member.getId().equals(id) && member.getPassword().equals(password)) {
            System.out.println(member.getName() + "dd");
            model.addAttribute("name", member.getName());
            memberinfo.setName(member.getName());
            return "redirect:/Home";
        }
    }
    model.addAttribute("error", "Please check your ID and password.");
    return "Login";
}
```

3. 홈 페이지

homePage(), homePage2() 메서드 : 상품을 조회하고, 해당 상품을 productList에 추가하여 홈 페이지에서 상품 목록을 표시하는 기능 구현

@RequestParam 어노테이션을 사용하여 HTTP 요청 파라미터를 매핑해 상품 코드를 파라미터로 받아서 해당 상품을 찾음 -> productList에 상품을 추가 -> productDao.selectAll()을 호출하여 데이터베이스에 저장된 모든 상품 정보를 가져옴 -> for문으로 가져온 상품 목록을 순회하면서 입력된 상품 코드와 일치하는 상품을 찾음 -> 일치하는 상품이 있으면, productList에 상품을 추가, 모델에 지갑 잔액(Wallet.getWallet()), 상품 목록(productList), 판매량(Sale.getSale()), 회원 이름(memberinfo.getName())을 추가

홈 페이지 템플릿("Home")을 반환하여 홈 페이지를 표시합니다.

```
List<Product> productList = new ArrayList<>();
@PostMapping(value = "/Home")
public String homePage(@RequestParam("productcode") String productcode, Model model) {
    List<Product> products = productDao.selectAll();

    for (Product product : products) {
        if (product.getCode().equals(productcode)) {
            System.out.println(product.getProductname());
            productList.add(product);
        }
    }
    model.addAttribute("wallet", Wallet.getWallet());
    model.addAttribute("products", productList);
    model.addAttribute("sale", String.valueOf(Sale.getSale()));
    model.addAttribute("name", String.valueOf(memberinfo.getName()));
    System.out.println(productList);
    return "Home";
}
```

새 상품 입고시 사용하는 SQL문

```
public void InsertSchedule(Object[] params) {

    String sql = "INSERT INTO Schedule(CODE, PRODUCTNAME, QUANTITY, PRICE, DATE) VALUES(?,?,?,?,?)";
    jdbcTemplate.update(sql, params);
}
```

4. 관리자 페이지

adminlogin() 메서드

@RequestParam 어노테이션을 사용하여 HTTP 요청 파라미터를 매핑해 비밀번호를 파라미터로 받아서 입력된 비밀번호와 pass 변수에 저장된 비밀번호를 비교

-> 비밀번호가 일치하면) "adminsucess"로 리디렉션

-> 비밀번호가 일치하지 않으면) 모델에 오류 메시지를 추가하고 "admin" 템플릿을 반환하여 관리자 로그인 페이지를 표시

```
@PostMapping(value = "/admin")
public String adminlogin(@RequestParam("password") String password, Model model) {
        if (password.equals(pass)) {
             return "redirect:/adminsucess";
        }
    model.addAttribute("error", "Please check your password.");
    return "admin";
}
```

adminsucess(), adminsucess2() 메서드

새로운 상품 정보를 받아서 데이터베이스에 저장하고, 모든 상품을 가져와서 productList2를 추가

-> 이 데이터를 바탕으로 일간, 주간, 월간, 판매량이 높은 상품들을 가져와서 모델에 추가

```
@PostMapping(value = "/adminsucess")
public String adminsucess2(@RequestParam("productcode") String productcode,
        @RequestParam("productname") String productname, @RequestParam("productquantity") long productquantity,
        @RequestParam("productprice") long productprice, Model model) {
    model.addAttribute("wallet", Wallet);
    LocalDateTime productdate = LocalDateTime.now();
    Object[] params = {productcode,productname,productquantity,productprice,productdate};
    productDao.InsertProduct(params);
    productList2.clear(); // productList 초기화

    List<Product> topSellingProducts = ScheduleDao.getTopSellingProducts("day");
    model.addAttribute("topSellingProducts", topSellingProducts);

    List<Product> weektopSellingProducts = ScheduleDao.getTopSellingProducts("week");
    model.addAttribute("weektopSellingProducts", weektopSellingProducts);

    List<Product> monthtopSellingProducts = ScheduleDao.getTopSellingProducts("month");
    model.addAttribute("monthtopSellingProducts", monthtopSellingProducts);


   List<Product> products = productDao.selectAll();
   for (Product product : products) {
           System.out.println(product.getProductname());
           productList2.add(product);
   }
   model.addAttribute("products", productList2);
   model.addAttribute("wallet", Wallet.getWallet());
   return "adminsucess";

}
```

productdelete()메서드

"delete" 매개변수로 삭제할 상품 코드를 받음 -> productDao.deleteProduct()메서드를 사용하여 해당 상품을 데이터베이스에서 삭제 -> "adminsucess" 페이지로 이동

```
@PostMapping(value = "/productdelete")
public String productdelete(@RequestParam("delete") String delete, Model model) {
        productDao.deleteProduct(delete);
        System.out.println(delete);

    return "redirect:/adminsucess";
}

public void deleteProduct(String productCode) {
    String sql = "DELETE FROM product WHERE CODE = ?";
    jdbcTemplate.update(sql, productCode);
}
```

qtyup2(), qtydown2() 메서드 : 관리자 페이지에서 상품 수량을 증가 또는 감소시키는 기능 처리

qtyup, qtydown 매개변수로 상품 코드를 받아서, productDao.increaseQuantity(), decreaseQuantity()메서드를 사용해 해당 상품의 수량을 1만큼 증가, 감소시키고 adminsucess페이지로 이동

```
@PostMapping(value = "/up")
public String qtyup2(@RequestParam("qtyup") String qtyup, Model model) {
    System.out.println(qtyup);

        productDao.increaseQuantity(qtyup, 1);


    return "redirect:/adminsucess";
}
@PostMapping(value = "/down")
public String qtydown2(@RequestParam("qtydown") String qtydown, Model model) {
    System.out.println(qtydown);

        productDao.decreaseQuantity(qtydown, 1);


    return "redirect:/adminsucess";
}
```

상품 수량 증가, 감소하는 것을 업데이트하는 SQL문

```
public void decreaseQuantity(String productCode, int quantity) {
    String sql = "UPDATE product SET QUANTITY = QUANTITY - ? WHERE CODE = ?";
    jdbcTemplate.update(sql, quantity, productCode);
}
public void increaseQuantity(String productCode, int quantity) {
    String sql = "UPDATE product SET QUANTITY = QUANTITY + ? WHERE CODE = ?";
    jdbcTemplate.update(sql, quantity, productCode);
}
```

일간, 주간, 월간 간격으로 통계자료 출력하는 SQL문

```
public List<Product> getTopSellingProducts(String duration) {
    String sql = "SELECT CODE, PRODUCTNAME, SUM(QUANTITY) AS TOTAL_QUANTITY, SUM(QUANTITY * PRICE) AS TOTAL_SALES FROM Schedule " +
                 "WHERE DATE >= ? " +
                 "GROUP BY CODE, PRODUCTNAME " +
                 "ORDER BY TOTAL_QUANTITY DESC " +
                 "LIMIT 5";

    LocalDateTime startDate;
    if (duration.equals("day")) {
        startDate = LocalDateTime.now().minusDays(1);
    } else if (duration.equals("week")) {
        startDate = LocalDateTime.now().minusWeeks(1);
    } else if (duration.equals("month")) {
        startDate = LocalDateTime.now().minusMonths(1);
    } else {
        throw new IllegalArgumentException("Invalid duration: " + duration);
    }
```

5. 상품 등록, 조회, 수정, 삭제, 결제

productcancel()메서드

"cancelcode" 매개변수로 취소할 상품 이름을 받음 -> productList에 저장된 상품들을 순회하면서 취소할 상품을 찾아 해당 상품을 productList에서 제거 -> "Home" 페이지로 이동

```
@PostMapping(value = "/cancel")
public String productcancel(@RequestParam("cancelcode") String cancelcode, Model model) {

    List<Product> products = productDao.selectAll();
    Iterator<Product> iterator = productList.iterator();

    while (iterator.hasNext()) {
        Product product = iterator.next();
        if (product.getProductname().equals(cancelcode)) {
            System.out.println(product.getProductname());
            iterator.remove();
        }
    }


    return "redirect:/Home";
}
```

상품 추가하는 SQL문

```
public void InsertProduct(Object[] params) {
    String sql = "INSERT INTO product(CODE, PRODUCTNAME, QUANTITY, PRICE, DATE) VALUES(?,?,?,?,?)";
    jdbcTemplate.update(sql, params);
}
```

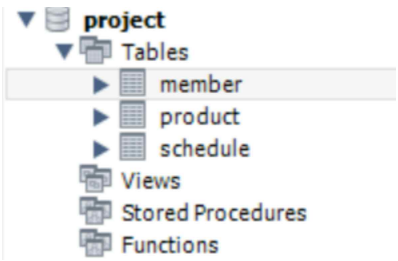
payment()메서드

"saleAmount" 매개변수로 판매 금액을 받음 -> productList에 저장된 상품들을 순회하면서 해당 상품의 수량을 1만큼 감소 -> 상품 코드, 상품 이름, 판매 수량, 상품 가격, 판매 일시를 배열로 저장해 판매 내역을 스케줄에 추가 -> 판매 금액을 계산하여 Sale에 저장하고, productList를 초기화 -> 현재 지갑 잔액에 'sale'값을 추가하고 -> 업데이트 된 값을 저장 및 출력 -> 초기 화면 "Home" 페이지로 리다이렉트

```
@PostMapping(value = "/payment")
public String payment(@RequestParam("saleAmount") long saleAmount, Model model) {
    for (Product product : productList) {
        productDao.decreaseQuantity(product.getCode(), 1);
        LocalDateTime productdate = LocalDateTime.now();
        Object[] params = {product.getCode(),product.getProductname(), 1 ,product.getPrice(), productdate};
        ScheduleDao.InsertSchedule(params);
    }
    long sale = saleAmount;
    productList.clear();
    model.addAttribute("sale", String.valueOf(Sale.getSale()));
    model.addAttribute("name", String.valueOf(memberinfo.getName()));
    Wallet.setWallet(Wallet.getWallet()+sale);
    model.addAttribute("wallet", Wallet.getWallet());
    System.out.println(Wallet.getWallet());
    System.out.println(Sale.getSale());

    return "Home";
}
```

MYSQL



Member Table

ID	PASSWORD	NAME	GRADE
admin	1234	Admin	Admin
wnjd	김주홍	1234	관리자
333	333	333	직원
admin	1234	김주홍	관리자
wnjd	1234	김주홍	관리자

Product Table

CODE	PRODUCTNAME	QUANTITY	PRICE	DATE
1	감자칩	7	2000	2023-06-12 17:08:01
10	피크닉청포도맛	30	1000	2023-06-13 15:49:58
2	우유500ml	17	2000	2023-06-13 02:07:30
3	초코파이12개입	3	5000	2023-06-13 02:07:46
4	Cass맥주	19	3000	2023-06-13 02:08:15
5	초코바	18	1000	2023-06-13 02:08:36
6	파워에이드	13	2000	2023-06-13 02:08:56
7	참이슬오리지널	17	1980	2023-06-13 02:09:38
8	삼각 김밥	10	1400	2023-06-13 15:49:26
9	실론티	30	1200	2023-06-13 15:49:42
NULL	NULL	NULL	NULL	NULL

Schdule Table

CODE	PRODUCTNAME	QUANTITY	PRICE	DATE
1	감자칩	1	2000	2023-06-13 02:10:37.648659
7	참이슬오리지널	1	1980	2023-06-13 02:10:37.656659
6	파워에이드	1	2000	2023-06-13 02:10:37.663948
2	우유500ml	1	2000	2023-06-13 02:10:37.672802
3	초코파이12개입	1	5000	2023-06-13 02:10:37.680569
1	감자칩	1	2000	2023-06-13 02:11:52.55005
1	감자칩	1	2000	2023-06-13 02:11:52.558283
2	우유500ml	1	2000	2023-06-13 02:11:52.597337
3	초코파이12개입	1	5000	2023-06-13 02:11:52.603931
3	초코파이12개입	1	5000	2023-06-13 02:11:52.61077
3	초코파이12개입	1	5000	2023-06-13 02:11:52.617317
3	초코파이12개입	1	5000	2023-06-13 02:11:52.624383
3	초코파이12개입	1	5000	2023-06-13 02:11:52.631318
5	초코바	1	1000	2023-06-13 02:11:52.63855
7	참이슬오리지널	1	1980	2023-06-13 02:11:52.644243
1	감자칩	1	2000	2023-06-13 02:27:14.518121
2	우유500ml	1	2000	2023-06-13 02:27:14.527833
1	감자칩	1	2000	2023-06-13 02:27:14.535395
3	초코파이12개입	1	5000	2023-06-13 02:27:14.543708
4	Cass맥주	1	3000	2023-06-13 02:27:14.55076
3	초코파이12개입	1	5000	2023-06-13 02:27:14.557869
5	초코바	1	1000	2023-06-13 02:27:14.564885
6	파워에이드	1	2000	2023-06-13 02:27:14.570573
7	참이슬오리지널	1	1980	2023-06-13 02:27:14.580125