기술 설명서

|  |  |
| --- | --- |
| **구분** | **내용** |
| **C (Insert,삽입)**  **(주문정보**  **입력 시 사용)** | Pom.xml에 프로그램 제작에 사용할 의존성, 기본설정값 등을 정의해놓는다.  JDBC(데이터베이스 조작), DBCP(데이터베이스 연결) 두 의존성을 기반으로 DB와 연동된다.  이러한 의존성은 java구문에 사용하기 위하여 application-config.xml파일에 네이밍과 설정값을 명시적으로 지정해주어 사용하고있다.  JDBC의 Property에 DBCP를 주어 JDBC를 Java에서 호출하여 사용하게 된다. JDBC는 jdbcTemplate로 네이밍했다.  book.jsp 의 POST방식의 form(bookForm)을 통해 BakeryController.java에 정의 된 @RequestParam을 통해 값을 받아 앞서 설명한 JDBC,DBCP를 이용하여 query문을 실행한다.  또한 매개변수로 book\_id는 입력해주지 않는데 이는oracle DB의 sequence 기능을 이용했다. Sequence는 유일한 값을 생성해주는 오라클 객체이며 처음 생성 시 입력한 설정값으로 증감값, 최대값, 시작값 등을 설정할 수 있다.  Book\_sequence로 네이밍 하였으며 시작값 1,증가값1의 속성을 주어 입력시 자동으로 book\_id(주문번호)가 입력되도록 했다.  (RequestParam은 HTML 요청 파라미터를 매핑한다.)  기존에 정의한 BookVO.class 의 상태값은 이와 동일한 형태의 Book table에 삽입된다. |
| **R (Select, 검색)**  **(주문검색 시**  **사용)** | JDBC, DBCP 의존성을 기반으로 DB와 연동하였으며 JDBC의 query기능을 사용하여 String형태의 쿼리문을 실행했다.  Book table(예약테이블)의 속성 중 phonenumber속성을 기준으로 검색을 하였으며 이는 lookbook\_2.jsp에서  c tag의 foreach구문을 통해 보여진다.  C tag는 JSP에서 사용되는 반복,조건,시간,문자열 가공 등에 사용되는 기능을 사용할 수 있게하는 태그이다.  @RequestParam 을 통해 bookNum으로 네이밍한 전화번호 입력란의 값을 받아오게 된다. 이는 BookRepository에 정의된 readBook(int)메소드에 전달되며 쿼리문을 통해 bookNum값과 같은 phonenumber을 가진 튜플을 BookVO로 반환받고 이를 ArrayList<BookVO>에 입력하여 반환한다.  Controller에서 반환받은 ArrayList<BookVO>는 Model객체에 bookVO란 이름으로 입력하여 프론트에 전달하게 된다.  전달받은 bookVO를 foreach구문을 통해 가시적으로 출력했다. |
| **U(Update,수정)**  **(주문 시 Bread 데이터 차감)** | ODBC에 저장되어 있는 Table을 DBCP를 통해 접속하고 JDBC를 통해 DB를 controll할 수 있는데 사용자가 빵을 예약하기 위해서는 Client에서 BakeryController.java로 @RequestParam으로 받을 파라미터 4개를 받는데 3개는 <form>태그를 이용해 넘겨주고 그 값으로는 “breadName” , “count” , “phoneNuber” 이다 1개는 JSP와 HTML간의 데이터 통신을 위한 Model으로서 3개의 input값을 Query 문을 통해 bread Table을 update하고 결과값을 Model.addAttribute를 사용해 값을 client로 보내준다. 이때 Client는 Model을 통해 받은 baykeryVO를 이용해 예약정보를 JS기능인 alert()를 통해 사용자에게 알려준다. |
| **D (Delete, 삭제)**  **(주문취소 시**  **사용)** | 삭제하기 위해서는 DBCP를 통해 DB의 정보를 가져오고 JDBC에서 Query문을 통해 제어를 할 수 있다. 사용자가 예약 정보를 알고 싶다면 textbox에 예약할 때 입력해둔 전화번호를 입력하면 client에서 form을 통해 phonrnumber라는 값을 controller로 보내준다. 이때 book(예약) Table의 index값을 알아야 삭제할 튜플을 삭제할 수 있기 때문에 Query문 where절에 rownum을 사용하여 table의 index값을 접근 하여 사용자가 삭제하고 싶은 index값을 입력을 하면 index값이 할당된 튜플이 삭제된다. |
| **페이지 이동** | 페이지 이동을 하기 위해 세미 컨트롤러가 client에서 요청 해줄 url를 @RequestMapping을 통해 미리 정의한다. Client는 JSP로 요청을 하는데 front\_controller가 Client에서 요청 받은 url과 url이 정의된 세미 컨트롤러를 호출한다. Client와 url의 값이 값다면 정의된 메서드가 실행되는데 예를 들면 client 에서 /lookbook으로 url를 호출하면 세미 controller에서 /lookbook으로 Mapping 되어있는 메서드를 호출하여 사용자에게 필요한 페이지를 보여주는 형식이다. |
| **UI적 기능** | home2.jsp는 div를 이용해 공간을 나눴다.  좌측 menu를 20% 우측 content를 80%로 나누었고 좌측 menu부분 a tag의 href를 이용해 화면전환을 구현했으며 이는 Controller의 RequestMapping기능을 통해 value값과 비교하여 자동 매핑된다.  매핑된 기능의 return 값을 통해 화면전환이 이루어진다. 또한 a태그 속성인 target에 content의 id를 입력하여 실질적인 화면 전환은 우측 content에서 동작하고 좌측의 menu는 유지되도록 구현했다.  Swiper라는 UI를 삽입했다 이는 이미지를 연결하여 슬라이드 형태로 구현해주는 오픈소스이다.  Swiper의 URL을 Link하여 불러왔다.  이는 main.jsp 에 구현했다.  Autoplay(자동재생), speed(재생속도), pagination(페이징 설정), clickable(클릭설정) 등을 기입하여 원하는 형태의 swiper를 구성할 수 있었다.  네비게이션 버튼을 생성해 next와 prev 버튼을 구성하여 클릭을 통해 다음사진과 이전 사진을 볼 수 있도록 구성하였다.  그리고 폰트 설정을 외부 폰트를 가져와 사용하는 방법을 사용했다. 방법은 구글에서 제공해준 <link>태그를 <head>태그에 정의를 해주고 정의된 폰트 이름을 가져왔다. 예를들어 <link>태그에 정의된 폰트 네임이 나눔 이라면 font-family “Oswald” 으로 html의 폰트 형식을 교체했다. |