쇼핑몰 프로젝트를 발표하겠습니다.

* (PPT 2페이지) 저는 스프링에 오픈 소스 빌드 툴인 메이븐, 배포 환경으로 아파치 톰캣을 사용하여 개발하였고 개발 환경, 프로젝트 설정 및 구조, 구현, 오류 해결 순으로 말씀드리겠습니다.
* (4페이지) 스프링 Plain Old Java Object 방식의 프레임워크 입니다. 특정한 인터페이스를 구현하거나 상속을 받을 필요가 없어 기존에 존재하는 라이브러리 등을 지원하기에 용이하고 객체가 가벼운 특징이 있습니다.
* 오픈 소스 빌드 툴인 메이븐은 컴파일과 빌드를 동시에 수행하고, 자동으로 스프링 라이브러리 기능 관리합니다.
* 아래는 메이븐 기반 웹 프로젝트의 기본 디렉터리 구조인데요,

최상위 프로젝트를 기준으로 pom.xml 이라는 메이븐 프로젝트 설정 파일이 존재하며 그것을 바탕으로 프로젝트와 관련된 정보를 기술하게 되어 있는 구조입니다.

스프링 배포 환경인 Apache Tomcat 에서, **아파치는 정적인 파일을 처리해 주는 웹 서버이고,** **톰캣은 DB처리와 같은 동적인 기능들을 가공하여 HTML파일로 만들어 클라이언트에게 제공하는 역할을 한다고 말씀드릴 수 있습니다.**

**서블릿은 main()메소드로 직접 호출되지 않고, 웹 컨테이너인 톰캣에 의해 실행되며 톰캣은 Servlet 클래스의 규칙에 맞게 서블릿 객체를 생성, 초기화, 호출, 종료하는 생명 주기를 관리하는 역할을 합니다.**

스프링의 핵심 요소는 크게 **IoC (Inversion of Control)**,**DI,AOP** 가 있습니다.

**IoC 는** '**제어의 역전**'이라고 하는데, 컨트롤의 제어권이 개발자에게 있는게 아니라 프레임워크에 있다는 것 입니다.

**DI (Dependency Injection)는**

**의존성 주입**이고, IoC 컨테이너에서 빈 객체를 생성하는 방식을 말합니다. 클래스에서 new 연산자 대신에 외부에서 주입하기 때문에 new연산자가 사라집니다. Spring Container에서 각 객체를 생성할 때는 단 한번만 실행되고, 이를 Singleton이라고 바꿔 말할 수 있습니다.

**AOP (Aspect Oriented Programing)는**

**관점 지향 프로그래밍**으로 로깅, 트랜잭션, 보안 등 여러 모듈에서 공통적으로 사용하는 기능을 분리하여 관리하는 것 입니다.

**(7페이지) Spring Container에 대해 말씀드리겠습니다.**

앞서 말씀드린 IoC와 DI의 원리가 적용되는데, 스프링은 스프링 컨테이너를 통해 객체를 관리합니다. **Spring Container는 Bean 생명 주기를 관리**하고, 생성된 자바 객체들에게 추가적인 기능을 제공하는 역할을 합니다.

(8페이지)**스프링 MVC의 구조에 대해 말씀드리겠습니다. 스프링 MVC는 Servlet Container 안에 있는** **Servlet 요소 중 하나로 보실 수 있습니다.**

(9페이지) 스프링 프레임워크는 MVC 패턴을 준수하는데,

M (Model) / V (View) / C (Controller)로 구성되어져 있는 구조입니다.

* MVC 패턴은 크게 MVC1패턴과 MVC2 패턴으로 나뉘는데,
* **MVC1** 패턴의 경우 View와 Controller를 모두 JSP가 담당하는 형태로, **JSP 하나에서 MVC 가 모두 이루어져 재사용성이 매우 떨어지고, 읽기 힘들어져 유지보수에 있어 문제가 발생합니다.**

**MVC2** 패턴은 Controller와 View가 분리되어 있고, 역할이 분리되어 MVC1패턴에서의 단점을 보완할 수 있습니다. 스프링은 **MVC2패턴을 채택하여** 유지보수에 있어 큰 이점을 가집니다 **.**

(10페이지)Spring MVC(Model-View-Controller)의 핵심 요소는 DispatcherServlet, HandlerMapping, Controller, ModelAndView, ViewResolver, View가 있는데,

|  |  |
| --- | --- |
| DispatcherServlet | 은 웹요청과 응답의 Life Cycle을 주관하고, |
| HandlerMapping | 은 웹요청시 해당 URL을 어떤 Controller가 처리할지 결정하며,. |
| Controller | 는 비지니스 로직을 수행하고 결과 데이터를 ModelAndView에 반영합니다. |
| ModelAndView | Controller가 수행 결과를 반영하는 Model 데이터 객체와 이동할 페이지 정보(또는 View객체)로 이루어져 있고, |
| ViewResolver | 어떤 View를 선택할지 결정합니다. |
| View | View에서는 모델데이터를 이용하여 결과 데이터인 Model 객체를 표시한다고 말씀드릴 수 있습니다. |

다음은 위 요소들의 흐름입니다.(그냥 참고)

유저의 요청을 받는 DispathcerServlet이 핵심이고, 브라우저에서 요청시 DispatcherServlet이 사용자의 요청을 받고, HandlerMapping으로 넘어갑니다.

HandlerMapping은 사용자의 요청과 매핑되는 Controller를 찾고, 요청 url에 해당하는 Controller 정보를 저장하는 table을 갖습니다.

즉, 클래스에 annotaion을 명시하면 해당 URL에 대한 요청이 들어왔을 때 table에 저장된 정보에 따라 해당 클래스 또는 메서드에 Mapping합니다.

Controller에서는http request를 처리해 모델을 만들고 뷰를 지정하고, **컨트롤러에** 의해 모델과 뷰가 래핑된 모델앤뷰가 반환됩니다.

ViewResolver는 Controller가 반환한 View Name에 prefix, suffix를 적용하여 View Object를 반환하고,

이 후에 View에서 모델데이터를 이용하여 결과 데이터인 Model 객체를 표시합니다.

프로젝트 설정 및 구조에 대해서 말씀드리겠습니다.

먼저 전체 DB구조입니다.

회원 정보 테이블은 회원 아이디를 키 값으로, 주문테이블은 주문상품 일련번호가 키값, 주문자아이디와 상품 번호가 외래키, 상품정보는 상품 번호가 키값으로 지정했으며, 장바구니테이블은 회원아이디와 상품번호를 외래키로 지정했습니다. 상품 이미지 정보 테이블은 이미지번호가 키값, 상품번호를 외래키로 지정했습니다.

주문 테이블은 회원 정보 테이블과 상품 정보 테이블을 동시에 참조하고 있고 장바구니 테이블은 회원ID와 상품 번호를 이용해 회원 정보 테이블과 상품 정보 테이블을 동시에 참조합니다. 상품과 상품 관련 이미지 정보는 상품 이미지 정보테이블에 따로 저장되는 구조입니다.

(14페이지)DB연동을 위해 마이바티스를 사용하였고,

마이바티스는 SQL문의 코드로부터 완전히 분리해 생산성, 유지보수성이 향상 되는 장점이 있습니다.

(15페이지)다음으로 프로젝트 기본 설정 및 구조에 대해서 말씀드리겠습니다.

src/main/java 에 개발되는 java 코드를 작성했고, Controller, Model, Service, Dao, VO등으로 구성했습니다.

서버가 실행될 때 필요한 파일들의 경로는 src/main/resources 에 mappers, tiles, models로 구성했습니다.

프로젝트의 전체 ROOT인 WebContent에서

Webapp의 resources에 (image file, css, js)등 정적인 데이터 파일을 넣었고, WEB-INF 에 Config, Erd, 라이브러리, 스프링, 뷰, web.xml 을 넣었습니다.

spring폴더에 Spring과 관련된 xml 설정 파일들을 작성하였고,

views에는 Controller와 매핑되는 .jsp 파일들을 작성했습니다.

web.xml은 web application의 설정을 위한 실제에 매핑 또는 URL 포함 서블릿 코드를 작성했습니다.

pom.xml에는 Servlet이나 JSP에서 추가로 사용하는 라이브러리 또는 드라이버를 추가해 구성하였습니다.

(16페이지)servlet-context.xml 입니다. 설정에 대해 말씀드리면~ ppt 읽기.

(17페이지) mybatis-context.xml입니다. mybatis-context.xml에서는 ~ppt 읽기.

(18페이지)pom 설정입니다. 웹 페이지에서 반복적으로 사용되는 부분들에 대한 정보를 모아서 관리해주는 타일즈, 데이터베이스에서 자료를 쿼리하거나 업데이트하는 방법을 제공하는 JDBC, DB연동을 위한 마이바티스, ajax를 이용해 검색 자동완성을 구현하기 위해 JSON 설정 등을 하였습니다.

(19페이지)Controller 계층 구조 및 Service와 DAO클래스의 계층 구조입니다.

모든 Controller 클래스들은 BaseController 클래스를 상속받고, ServiceImpl과 Service, DAOImpl과 DAO 간에는 구현하는 구조입니다.

(20페이지)구현에 대해 말씀드리겠습니다.

(21페이지)메인 페이지 입니다.

Header, side, footer, quickmenu, main 으로 나누어 작성하고 관리하였습니다.

(22페이지) **tiles\_main.xml**에 레이아웃을 지정하는 JSP파일 위치를 지정하여 출력이 되게 하였습니다.

**MainController** 에서 **\*\*\*ControllerImpl 클래스 브라우저에서 매핑 된 주소로 요청 시 ModelAndView 객체에 설정 된 뷰 이름을 tilesViewResolver로 반환하도록 작성하였습니다.**

(23페이지) 메인 페이지에 표시할 상품을 조회하는 SQL문을 매퍼파일인 **goods.xml에 작성하였고, 상품 정보를 저장하는 resultMap, 상품 이미지를 저장하는 resultMap, #{goods\_status}의 조건 값으로 상품을 최대 15개까지 조회하는 sql 문을 작성하였습니다.**

(24페이지)

**GoodsDAOImpl 클래스에서는 전달된 조건으로 DB의 도서 정보를 조회하고,**

GoodsServiceImpl 클래스에서는 베스트셀러, 신간, 스테디셀러를 sql문의 조건으로 전달하여 각각의 도서 정보를 조회해서 hashmap에 저장한 후 컨트롤러로 반환합니다. Maincontroller 클래스에서는 브라우저 요청 시 베스트셀러, 신간, 스테디셀러를 조회한 후, map에 저장하여 jsp로 전달 합니다.

(파란네모박스는 다음 장 동영상 재생하면서)

**사이드는 기본, 마이페이지, 관리자 메뉴 이렇게 나뉘어서 다르게 나오는데**

**(24페이지로 돌아와서 파란네모박스) setAttribute로 이름이 ‘side\_menu’인 객체 생성하여 ModelAndView로 저장 → side.jsp에서 셋팅된 속성 화면에 구현**

**합니다.**

**(26페이지)**

**상품 상세 페이지와 상품 조회 기능 구현에 대해 말씀드리겠습니다.**

**(동영상)메인페이지에서 상품을 누르면 상품 상세 페이지가 나오고 퀵메뉴에 조회한 상품이 추가됩니다.**

**(27페이지) Goods.xml에 상품 상세 페이지에 표시한 상품 정보와 상품 메인 이미지 및 상세 이미지 정보를 조회하는 sql문을 추가하였고,**

**GoodsServiceImpl 클래스에 상품 정보와 이미지 정보를 조회한 후 HashMap에 저장하고, GoodsControllerImpl 클래스에 조회한 상품 정보를 빠른 메뉴에 표시하기 위해 전달했습니다.**

**(28페이지) quickMenu.jsp에서는 동일한 <hidden> 태그에 연속해서 저장하면 배열로 저장하고 퀵 메뉴에 다음 클릭 시 <hidden> 태그에 저장된 상품**

**정보를 가져와 이미지 표시하게 하였습니다.**

**(29페이지-동영상 재생) 검색 자동 완성 기능입니다.**

**검색 자동 완성 기능은 ajax를 이용하여 키워드를 입력했을 때 관련 키워드를 자동으로 표시하도록 하였습니다.**

**(30페이지) goods.xml의 sql문에 like연산자를 사용해 검색창에 입력한 키워드를 가져와 그 키워드가 포함된 상품 제목을 조회하고, ajax로 브라우저에서 전송된 키워드를 가져와 그 키워드가 포함된 제목 목록을 json으로 만들어 다시 브라우저로 전송합니다.**

**(31페이지) 사용자가 검색 키워드를 입력하면 ajax기능을 이용해 해당 키워드가 포함된 목록을 가져오고, id가 suggest인 div태그에 차례로 표시합니다.**

**(32페이지)**

**MemberControllerImpl 클래스에서 상품 주문 과정시 로그인 했으면 주문 진행되고, 그 외에는 메인 페이지가 표시되도록 구현하였습니다.**

**HEADER.JSP에 로그인 하지 않은 상태, 로그인하였을 때, 관리자로 로그인 하였을 때의 header 부분이 다르게 구현하였습니다.**

**(33페이지)회원가입기능은 member.xml에 ID 중복 검사, 가입창에서 입력한 회원 정보를 테이블에 추가하는 SQL 문을 작성하였습니다.**

**다음에서 제공하는 주소 검색을 사용하기 위해 포함하였고, 지번 주소와 도로명 주소를 텍스트 창에 표시하도록 하였습니다.**

**(34페이지)장바구니 기능입니다. 장바구니 테이블 추가하기 전 그 상품 번호가 장바구니에 담겨있는지 조회하고, 상품을 추가하는 SQL문을 CART.XML에 작성하였습니다. CartControllerImpl 클래스에서 상품 번호가 이미 장바구니 테이블에 있으면 이미 추가되었다는 메시지를 브라우저로 전송하고, 없으면 테이블에 추가하도록 하였습니다.**

**(35페이지)CartServiceImpl 클래스에서 테이블에 추가하기 전에 동일한 상품 번호의개수를 조회하고 장바구니에 추가하도록 작성 하였고, CartDAOImpl 클래스는 이미 장바구니에 추가된 상품인지 조회하고 장바구니에 추가하도록 작성하였습니다.**

**로그인 상태로 장바구니에 물건이 없으면 담고, 이미 등록된 상품이면 등록된 상품이라는 메시지가 나옵니다.**

**(36페이지)주문 처리입니다. 로그인 여부에 따른 주문 페이지 처리를 하도록 구현하였는데, Order.xml에 <selectKey> 태그를 이용해 오라클의 시퀀스를 번호를 가져온 다음 가져온 다음 주문테이블의 각 레코드 구분 번호로 사용하였습니다.**

**주문번호는 주문 상품의 개수와 관계없이 주문 당 한 개로 생성하였으며, tiles\_order.xml에 장바구니에서 한 개의 상품을 주문할 경우 주문페이지 표시,**

**장바구니에 담긴 모든 상품을 주문할 경우 주문페이지 표시, 주문 결과 표시하도록 지정하였습니다.**

**(37페이지) 제품 상세페이지에서 바로 주문하는 기능에 대해 말씀드리겠습니다. 로그인 여부에 따라 달라지는데 로그인 했을 경우에는 주문하기를 클릭하면 브라우저에서 전송된 상품 정보를 arrayList에 저장해 세션에 바인딩 한 후 주문 페이지로 이동하고 주문 페이지에서 수령자 정보와 배송지 정보를 입력 받습니다. 최종 주문 시 컨트롤러에서 미리 세션에 저장된 주문 상품 목록을 가져와 전송된 수령자 정보와 배송지 정보를 합칩니다. 주문 정보를 합친 최종 arrayList를 sql문으로 전달하여 주문을 처리합니다.**

**OrderController 클래스에 로그인 여부에 따른 주문 페이지 처리 및 주문창 전달, 수령자 정보 설정 후 myOrderList저장 후 주문 완료 결과창에 주문자 정보 및 상품 목록을 표시하도록 전달하도록 작성하였습니다.**

**(38페이지)orderGoodsForm.jsp에는 상품 주문 페이지가 표시되면 주문자의 휴대폰 번호와 유선 전화번호를 셀렉트 박스를 표시하고 최종 결제하기 클릭시 주문창에서 입력한 수령자 정보를 설정하여 컨트롤러로 전송하도록 작성하였습니다.**

**(39페이지 –동영상 재생) 장바구니에 담긴 상품들을 한꺼번에 주문하는 기능에 대해 말씀드리겠습니다. 장바구니 페이지에서 주문할 상품을 선택한 후, 주문 수량을 입력하고 주문하기를 클릭하면 주문확인 페이지에는 선택한 주문 상품만 표시합니다.**

**(40페이지)OrderControllerImpl 클래스에 전송된 상품 번호와 GoodsVO의 상품 번호가 같으면 주문하는 상품이므로 OrderVO객체를 생성한 후**

**상품 정보를 OrderVO에 설정하고 다시 myOrderList에 저장하고, 선택한 상품만 myOrderList에 저장한 후 세션에 바인딩하도록 작성하였습니다.**

**(41페이지) 상품 하나만 주문할 경우 문자열로 전송하고, 여러 상품을 주문할 경우 하나의 상품에 대해 ‘상품 번호:주문수량‘ 문자열을 만든 후 전체 상품 정보를 배열로 전송하도록 작성하였습니다.**

**(42페이지) 결제 기능에 대해 말씀드리겠습니다.** **결제는 수기로 입력하는 방식으로 구현하였는데, REST API, HTTP 통신 기능을 제공하는 OkHttp 와 API에서 작동 및 일반 데이터 바인딩 기능이 있는 jackson-databind를 pom.xml에 추가하고,** **매퍼파일인** **tiles\_order.xml에 상태에 해당하는 파일로 연결되도록 작성하였습니다.**

**(43페이지)** ApiService 클래스에서는 모바일 데이터를 파싱해 map형태로 저장하도록 작성하였고, **TestController클래스에서는 파라이터로 사용할 맵을 작성하고, 사용자가 작성한 내용과 주문번호, 가격,** apiCertKey, timestamp 을 더한 값인 signaturer값을 해시맵 형태로 resultMap에 저장되어 반환하도록 작성하였습니다.

**(44페이지 –동영상 재생) orderGoodsForm의 fn\_show\_order\_detail 함수는 카카오페이 ajax 동기식 처리를 하는 코드를 작성하였고,**

**Fn\_process\_pay\_order함수에서는 결제정보를 체크하는 조건문을 작성하였습니다.**

**(45페이지) 마이페이지 메인 화면 구현에 대해서 말씀드리겠습니다. 주문 당 주문 상품명이 한꺼번에 나오도록 구현하였는데, mypage.xml에는 회원이 주문한 상품 목록을 표시하는 sql문을 작성하였고, MyPageControllerImpl 클래스에 회원 아이디를 이용해 주문 상품을 조회하고 주문 상품 목록을 jsp로 전달하는 코드를 작성하였습니다.**

(46페이지) 관리자가 각 주문 상품의 배송 상태를 설정하거나 변경하는 기능을 구현하는 기능에 대해 말씀드리겠습니다.

표는 쇼핑몰 주문 테이블의 배송 상태를 나타내는 delivery\_state 칼럼의 값들입니다.

admin\_order.xml에 주문 상품 목록을 표시하고, update문을 이용해 주문 상품의 배송 상태를 수정하는 sql문을 작성하였습니다

(47페이지) **AdminOrderControllerImpl 클래스에 Ajax로 주문 번호와 배송 설정 상태 값을 전달받아 sql문으로 전달하도록 작성했습니다.**

(48페이지) **adminOrderMain.jsp에서 배송 수정을 클릭하면 선택한 셀렉트 박스 id를 함수로 전달해 함수로 전달받은 셀렉트 박스에서 선택한 옵션의 인덱스로 선택한 배송 상태 값을 가져오도록 작성하였습니다.**

**오른쪽은 관리자로 로그인 했을 때, 배송 상태를 변경하고 적용한 후, 일반 사용자로 로그인 하였을 때 상태가 변경되는 모습입니다.**

**(49페이지)ppt를 작성하면서 도움을 받은 사이트들입니다.**

**(50페이지) 마지막으로 오류 해결에 대해 말씀드리겠습니다.**

**회원정보에 있는 null 값으로 인해 관리자 모드에서 회원 관리 페이지가 출력되지 않는 오류가 있었습니다. orale null 처리에 필요한 셋팅 프로퍼티 추가해 정상적으로 출력되게 하였습니다.**

**이상입니다. 감사합니다.**