<도연**>**

1. **SellController** 클래스

- **getList() :** 전체 **SellItem** 리스트를 가져오는 메소드
  SellServiceImpl.getSellList() 메소드를 호출해서 전체 SellItem 리스트를 가져온 후
  return (ModelAndView mav = new ModelAndView("sell/sell_list");
  mav.addAllobjects(list);)

- **getItem() :** 판매 리스트에서 각 판매 아이콘 클릭 시 세부 판매글로 이동하는 메소드
  SellServiceImpl.getMemberIdbyItemId() -> memberId 얻고 그거랑 세션의 memberId랑 비교, 둘이 같으면 판매자 페이지 호출, 둘이 다르면 구매자 페이지 호출, SellServiceImpl.getSellByItemId() -> sellItem 객체 얻어옴
  SellServiceImpl.getCountbyId() -> id 통해 tradeCount 값 얻어온 후 sellItem 객체에 set

- **searchList() :** 책 이름을 통해 판매 게시글 검색 후**,** 검색 결과 리스트를 보여주기 위한 메소드
- **getHistory() :** 세션에 저장된 **memberId**를 이용해 해당 회원이 판매한 전체 아이템 리스트를 보여주기 위한 메소드
- **upload() :** 판매글 등록 메소드
  requestParameter 통해 insert할 SellItem 객체 생성 -> deal_status는 0으로 초기화
  SellServiceImpl.insertSellItem() 통해 insert 작업 후 SellServiceImpl.getSellList() 통해 다시 SellItem 리스트 가져와서 sell_list.jsp로 이동
- **delete() : itemId** 이용해 판매글 삭제 메소드
  삭제 후 SellServiceImpl.getSellList()를 통해 전체 SellItem 리스트를 가져온 후 return
- **chat() :** 채팅 메시지로 이동하는 메소드
  SellServiceImpl.getDealStatus() 호출 -> 값이 0이 아니면
  SellServiceImpl.getSellByItemId(), SellerviceImpl.getCountbyId() 호출해서 SellItem 객체 생성 후 구매자에서 페이지로 이동, 경고 알림 띄움
  값이 0이면 -> SellServiceImpl.getMemberIdBySell() 통해서 판매자 아이디 얻은 후, itemId와 함께 ChatMessage 객체에 담아서 return

  @SessionAttributes("StatusForm")

2. **SellStatusController** 클래스

- **buyFin() :** 세션에 저장된 **memberId**를 이용해 해당 회원이 주문 완료한 전체 아이템 리스트를 보여주기 위한 메소드
- **formData() : StatusForm** 객체 생성 및 초기화 **-> itemId set, memberId set**
- **showForm() : itemId** 가지고 거래 상태 수정 페이지로 이동
- **update() :** 거래 상태 수정 메소드
  StatusForm 객체에 deal_status 정보 담아서 SellServiceImpl.updateDealStatus() 호출
  if id 파라미터가 넘어왔으면
  SellServiceImpl.getMembeIdById() -> 이름으로 아이디 구하기
  SellServiceImpl.getSellByItemId() -> 아이템 아이디로 BuyFin 객체에 넣을 SellItem 객체 가져옴
  가져온 SellItem 객체로 BuyFin 객체 생성
  SellServiceImpl.insertBuyFin() - 주문 완료 등록

공통 : SellServiceImpl.getSellByItemId() -> SellItem 객체 return

3. **StatusForm** 클래스 **//Command** 클래스

```
public class StatusForm {
        private int sell_itemId;
        private int deal_status;
        private int memberId;
        private int tradeCount;

        //getter, setter
}
```

@SessionAttributes(“updateForm”)

4. **SellUpdateController**클래스

- **formData() : UpdateForm** 객체 생성 및 초기화 **-> itemId set, memberId set**
- **showForm() : itemId** 가지고 판매글 수정 페이지로 이동
  SellServiceImpl.getDealStatus() 호출 -> 값이 0이 아니면
  SellServiceImpl.getSellByItemId(), SellerviceImpl.getCountbyId() 호출해서 SellItem 객체 생성 후 구매자에서 페이지로 이동, 경고 알림 띄움
- **update() :** 판매글 수정 메소드
  수정 후 SellServiceImpl.getSellByItemId(), SellerviceImpl.getCountbyId()를 통해 SellItem 객체 생성 후 return

5. **UpdateForm** 클래스 **//Command** 클래스

```
public class StatusForm {
        private int sell_itemId;
        private String sell_itemName;
        private int sell_price;
        private String sell_author;
        private String sell_publisher;
        private String sell_info;
        private String sell_image;
        private String deal_method;
        private int memberId; //판매자의 아이디

        //getter, setter
}
```

6. **WishController**클래스

- **getList() : List<Wishlist>** 객체를 가지고 즐겨찾기 조회 페이지로 이동하는 메소드
- **upload() :** 즐겨찾기 등록 메소드
  넘어온 파라미터를 통해 Wishlist 객체 생성 후 SellServiceImpl.insertWishlist() 호출
  수정 후 SellServiceImpl.getSellByItemId(), SellerviceImpl.getCountbyId()를 통해 SellItem 객체 생성 후 return
- **delete() :** 즐겨찾기 삭제 메소드
  삭제 후 SellServiceImpl.selectWishlist()를 통해 전체 Wishlist 리스트를 가져온 후 return

<지우**>** 로그인 및 회원가입 **/** 스터디원 모집 **/** 채팅 구현

1. **MemberController** 클래스

- **login() :** 로그인 페이지로 이동하는 메소드
    - 사용자가 로그인 한 정보를 가지고 있는 경우**, main** 페이지로 이동한다**.**

- **loginDo() :** 로그인 처리를 하는 메소드 **(**검증**)**
    - 넘어온 파라미터를 가지고**, DB** 로 부터 유효한 회원인지 검증 한다**.**

- **register() :** 회원가입 페이지로 이동하는 메소드

- **registerDo() :** 회원가입 처리 하는 메서드
    - 넘어온 파라미터를 가지고**, DB** 에 회원 정보를 등록한다**.**

2. **StudyController** 클래스

- **study() :** 스터디 페이지로 이동하는 메서드
- **studyRegister() :** 스터디 모임 등록하는 페이지로 이동하는 메서드
- **studyRegisterDo() :** 스터디 모임 등록하는 메서드
    - 넘어온 파라미터를 가지고**,** 스터디 모임을 등록한다**.**
- **studyJoinDo() :** 스터디 참가 하는 메서드
    - 채팅 페이지로 이동한다**.**

3. **ChatController** 클래스

- **chat() :** 채팅 페이지로 이동하는 메서드
- **chatMessageDo() :** 채팅 메시지 입력 메서드
    - **DB** 에 채팅 메시지 정보를 저장한다**.**

<하윤**>** 공동구매 기능 총괄 **/** 책 **API**

1. **GroupPurchaseApplierController** 클래스 **:** 공동구매 신청에 관련된 컨트롤러 클래스

- **apply() //** 공동구매 신청 **(**세션에서 아이디 정보 가져온다 **->** 누가 어떤 게시글에서 신청했는지**)**
- **cancel() //** 공동구매 신청취소

2. **GroupPurchaseController** 클래스 **:** 공동구매 페이지에서의 기본작업

- **getPostList() //** 전체 게시글 불러오기
- **detailPost() //** 게시글 자세히 보기
- **searchPost() //** 게시글 검색

3. **GroupPurchaseLeaderController :** 공동구매 게시글 작성자**(**총대**)**

- **uploadPost() //** 공동구매게시글 등록**,** 입력

4. **BookController** 클래스

- **Serach_BookName() //** 책 제목으로 찾기
- **Serach_BookId() // book id(ISBN)**으로 찾기
- **Resister_Book() //** 책 정보 등록**(**가져오기**)**

<지운**>** 경매 기능 총괄 **/** 책 **API**

1. **AuctionController :** 가장 기본적인 경매 컨트롤러**.**
   -getAuctionList() : 경매 아이템 게시글 목록 불러오는 메소드**.**
   -searchAuctionItem(String au_itemName) : 경매 아이템 검색할 때 사용하는 메소드

2. **AuctionPostController :** 게시글 업로드시 사용하는 컨트롤러
   - uploadPost() : 경매 게시글 업로드할 때 사용**.**
   - getAuctionPost() : 경매 게시글 자세히 불러올 때 사용**.**

3. **AuctionPostUpdateController :** 게시글 수정할 때 사용하는 컨트롤러

- **updatePostForm() :** 경매 게시글 수정하기 검색하기 눌렀을 때 사용하는 메소드
- **updatePost() :** 경매 게시글 수정했을 때 사용하는 메소트

4. **AuctionBidController :** 경매 입찰 시 사용하는 컨트롤러
   -insertAuctionBid() : 경매 게시글에 입찰하고 싶을 때 쓰는 메소드
   -closeAuctionBId() : 경매 게시글 마감**(**낙찰**)**시 사용하는 메소드**.**