

< 분산 프로그래밍 >

기말 REPORT

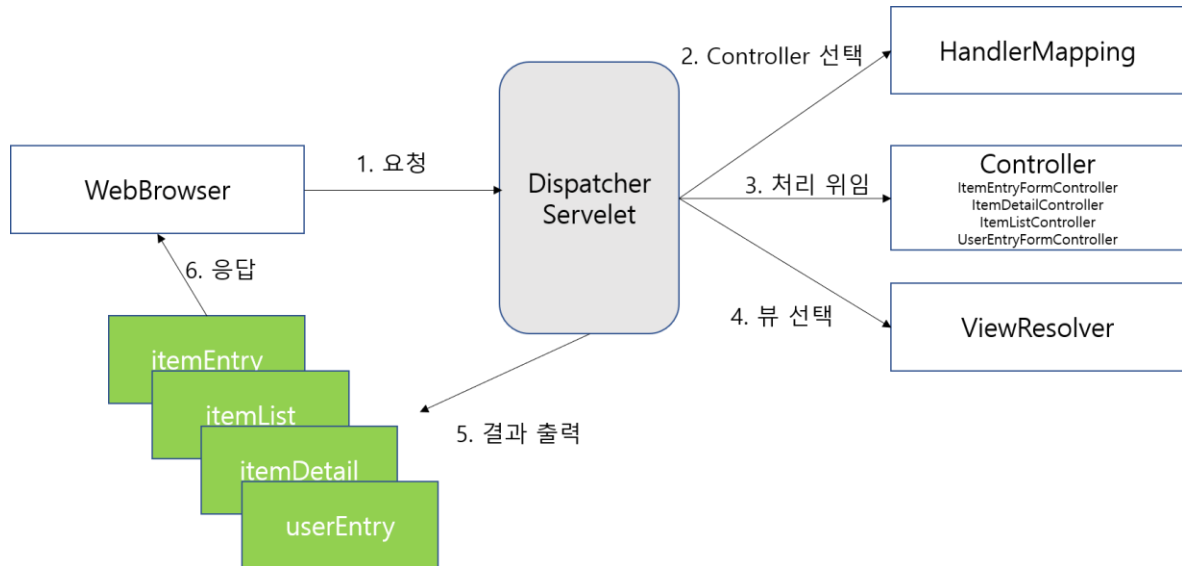
목차

1. 설계	1
1) SpringMVC	
2) UseCase	
2. 코드	4
1) 코드리스트	
2) 코드코멘트	
3) 실행결과	

과목명	분산프로그래밍
교수명	김정호
학과	융합소프트웨어학부
학번	60161606
이름	박준현

1. 설계

1) SpringMVC



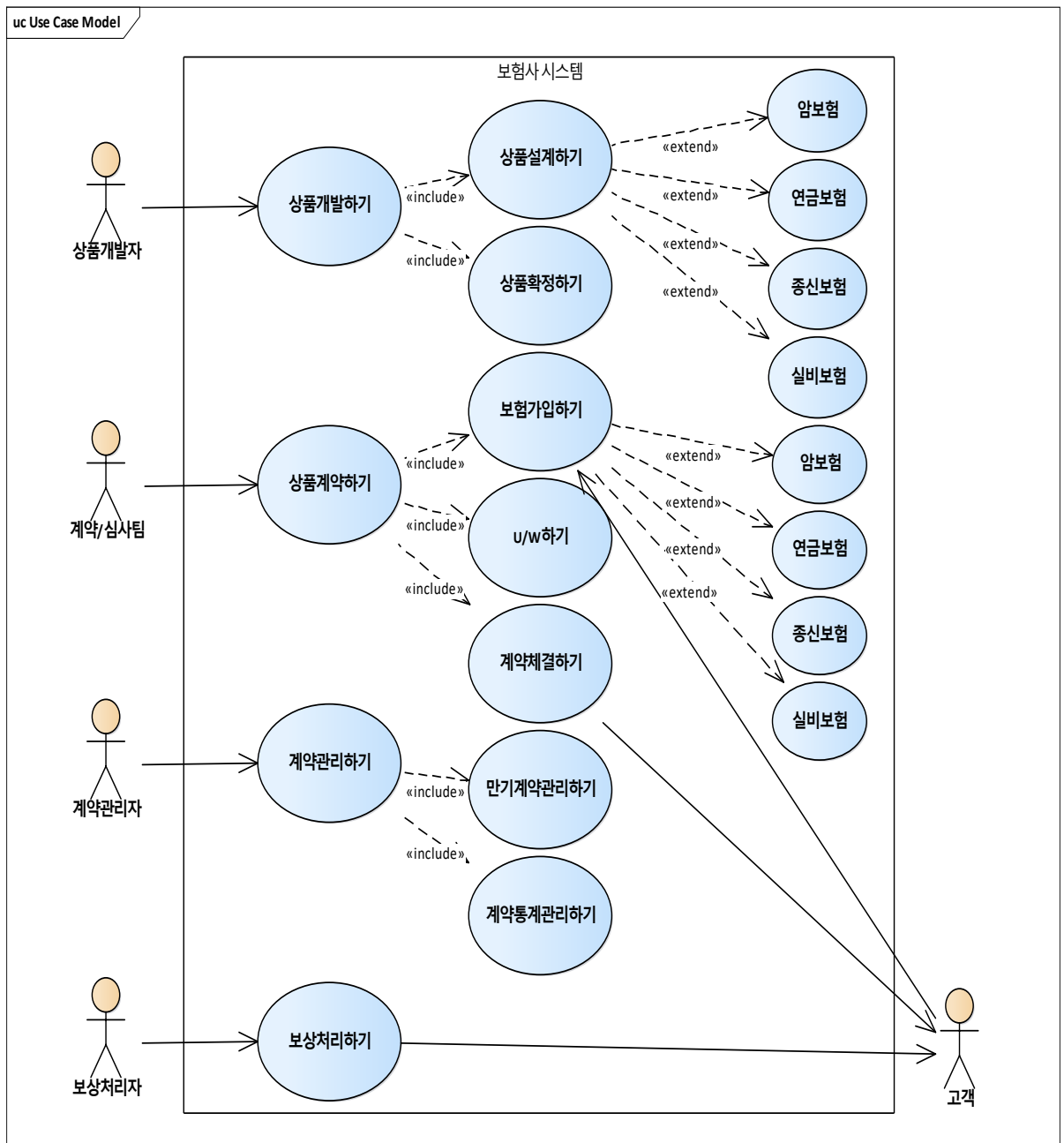
- 클라이언트에서 URL로 접근하여 DispatcherServlet에 정보를 요청
- DispatcherServlet은 HandlerMapping을 통해 요청을 매핑한 컨트롤러가 있는지 확인
- 해당 Controller를 선택하여 처리요청을 하고 결과를 출력할 View의 이름을 Return
- Controller에서 보내온 View이름을 토대로 View를 ViewResolver에서 검색
- ViewResolver의 처리 결과를 View에 출력하고 처리 결과가 포함된 View를 다시 Dispatcher에 송신
- Dispatcher는 최종 결과를 클라이언트에 출력

<Controller별 역할>

- ItemEntryFormController : 보험상품 입력을 위한 Controller (상품설계하기 scenario)로 새로운 보험을 설계할 수 있다.
- ItemListController : 보험상품 확정을 위한 Controller (상품확정하기 scenario)로 생성된 보험들의 확정을 결정할 수 있다.

- ItemDetailController : 보험상품의 세부정보를 보여주기 위한 Controller로 상품가입으로 넘어갈 수 있다.
- UserEntryFormController : 보험상품의 가입을 위한 Controller (가입하기 scenario)로 확정된 보험들을 가입자 정보를 입력하여 가입할 수 있다

2) UseCase 시나리오



- Java 코드로 구현된 보험사 시스템
- 1. 보험 설계하기
- 2. 보험 확정하기
- 3. 보험 가입하기
- 4. 보험 UW하기
- 5. 보험 계약하기
- 6. 보상 처리하기
- SpringMVC로 옮겨진 보험사 시스템
- 1. 보험 설계하기
- 2. 보험 확정하기
- 3. 보험 가입하기

2. 코드

1) 코드리스트

- input.controller → ItemEntryFormController, ItemListController, ItemDetailController, UserEntryController
- input.dao → ItemDao, ItemDaoImpl, UserDao, UserDaoImpl
- input.logic → Item, User, ItemCatalog, ItemCatalogImpl, UserCatalog, UserCatalogImpl, Shop, ShopImpl
- input.util → ItemValidator, UserValidator

2) 코드코멘트

Servlet-context.xml

```
7 <!-- HandlerMapping -->
8 <bean id="handlerMapping"
9     class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
10     <property name="mappings">
11         <value>
12             /itemEntry.html=itemEntryFormController
13             /itemList.html=itemListController
14             /itemDetail.html=itemDetailController
15             /userEntry.html=userEntryFormController
16         </value>
17     </property>
18 </bean>
```

HandlerMapping을 통해 View별 Controller 배정

```
20 <!-- Controller -->
21 <bean id="itemEntryFormController" class="com.spring.input.controller.ItemEntryFormController">
22     <property name="shopService" ref="shopService">
23         <value>shopService</value>
24     </property>
25     <property name="messageSource" ref="messageSource">
26         <value>messageSource</value>
27     </property>
28 </bean>
29 <bean id="userEntryFormController" class="com.spring.input.controller.UserEntryFormController">
30     <property name="shopService" ref="shopService">
31         <value>shopService</value>
32     </property>
33     <property name="messageSource" ref="messageSource">
34         <value>messageSource</value>
35     </property>
36 </bean>
37 <bean id="itemListController" class="com.spring.input.controller.ItemListController">
38     <property name="shopService" ref="shopService">
39         <value>shopService</value>
40     </property>
41 </bean>
42 <bean id="itemDetailController" class="com.spring.input.controller.ItemDetailController">
43     <property name="shopService" ref="shopService">
44         <value>shopService</value>
45     </property>
46 </bean>
47 <bean id="shopService" class="com.spring.input.logic.ShopImpl">
48     <property name="itemCatalog" ref="itemCatalog">
49         <value>itemCatalog</value>
50     </property>
51     <property name="userCatalog" ref="userCatalog">
52         <value>userCatalog</value>
53     </property>
54 </bean>
55 <bean id="itemEntryValidator" class="com.spring.input.utils.ItemEntryValidator">
56 </bean>
57 <bean id="userEntryValidator" class="com.spring.input.utils.UserEntryValidator">
58 </bean>
59 <bean id="messageSource"
60     class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
61     <property name="basenames">
62         <list>
63             <value>resources/messages</value>
64         </list>
65     </property>
66 </bean>
67 <bean class="com.spring.input.logic.UserCatalogImpl" id="userCatalog">
68     <property name="userDao" ref="userDao">
69         <value>userDao</value>
70     </property>
71 </bean>
72 <bean class="com.spring.input.dao.UserDaoImpl" id="userDao">
73 </bean>
74 <bean class="com.spring.input.logic.ItemCatalogImpl" id="itemCatalog">
75     <property name="itemDao" ref="itemDao">
76         <value>itemDao</value>
77     </property>
78 </bean>
79 <bean class="com.spring.input.dao.ItemDaoImpl" id="itemDao">
80 </bean>
```

Line: 29

각 class의 bean생성

```
62 <!-- ViewResolver -->
63 <bean id="internalResourceViewResolver"
64     class="org.springframework.web.servlet.view.InternalResourceViewResolver">
65     <property name="viewClass">
66         <value>org.springframework.web.servlet.view.JstlView</value>
67     </property>
68     <property name="prefix">
69         <value>/WEB-INF/views</value>
70     </property>
71     <property name="suffix">
72         <value>.jsp</value>
73     </property>
74 </bean>
75 </beans>
```

ItemEntryFormController

```
1 package com.spring.input.controller;
2
3 import org.springframework.context.MessageSource;
4
5 @Controller
6 public class ItemEntryFormController {
7
8     private ItemEntryValidator itemEntryValidator;
9     private Shop shopService;
10    private MessageSource messageSource;
11
12    public void setItemEntryValidator(ItemEntryValidator itemEntryValidator) {
13        this.itemEntryValidator = itemEntryValidator;
14    }
15    public void setShopService(Shop shop) {
16        this.shopService = shop;
17    }
18    public void setMessageSource(MessageSource messageSource) {
19        this.messageSource = messageSource;
20    }
21
22    @RequestMapping(method = RequestMethod.GET)
23    public String toItemEntryView() {
24        return "itemEntry";
25    }
26
27    @RequestMapping(method = RequestMethod.POST)
28    public ModelAndView onSubmit(Item item, BindingResult bindingResult) throws Exception {
29        ModelAndView modelAndView = new ModelAndView();
30        if (bindingResult.hasErrors()) {
31            modelAndView.getModel().putAll(bindingResult.getModel());
32            return modelAndView;
33        }
34        try {
35            this.shopService.entryItem(item);
36            modelAndView.setViewName("itemEntrySuccess");
37            modelAndView.addObject("item", item);
38            return modelAndView;
39        } catch (Exception e) {
40            // 상품ID가 중복일 때, 품종 중복일 때
41            bindingResult.reject("error_duplicate_list");
42            modelAndView.getModel().putAll(bindingResult.getModel());
43            return modelAndView;
44        }
45    }
46}
```

정보입력시 Validator가 입력정보 판독

입력이 성공적으로 완료되면 itemEntrySuccess View 띄움

UserEntryFormController

```
1 package com.spring.input.controller;
2
3 import org.springframework.context.MessageSource;
4
5 @Controller
6 public class UserEntryFormController {
7
8     private UserEntryValidator userEntryValidator;
9     private Shop shopService;
10    private MessageSource messageSource;
11
12    public void setUserEntryValidator(UserEntryValidator userEntryValidator) {
13        this.userEntryValidator = userEntryValidator;
14    }
15    public void setShopService(Shop shop) {
16        this.shopService = shop;
17    }
18    public void setMessageSource(MessageSource messageSource) {
19        this.messageSource = messageSource;
20    }
21
22    @RequestMapping(method = RequestMethod.GET)
23    public String toUserEntryView() {
24        return "userEntry";
25    }
26
27    @RequestMapping(method = RequestMethod.POST)
28    public ModelAndView onSubmit(User user, BindingResult bindingResult) throws Exception {
29        ModelAndView modelAndView = new ModelAndView();
30        if (bindingResult.hasErrors()) {
31            modelAndView.getModel().putAll(bindingResult.getModel());
32            return modelAndView;
33        }
34        try {
35            this.shopService.entryUser(user);
36            modelAndView.setViewName("userEntrySuccess");
37            modelAndView.addObject("user", user);
38            return modelAndView;
39        } catch (Exception e) {
40            // 유저ID가 중복일 때, 품종 중복일 때
41            bindingResult.reject("error_duplicate_user");
42            modelAndView.getModel().putAll(bindingResult.getModel());
43            return modelAndView;
44        }
45    }
46}
```

정보입력시 Validator가 입력정보 판독

입력이 성공적으로 완료되면 userEntrySuccess View 띄움

ItemDetailController

```
1 package com.spring.input.controller;
2
3 import java.util.HashMap;
4
5 @Controller
6 public class ItemDetailController {
7     private Shop shopService;
8
9     public void setShopService(Shop shop) {
10         this.shopService = shop;
11     }
12
13     @RequestMapping
14     public ModelAndView detailItem(String itemId){
15         // 선택된 상품ID에서 상품 정보를 가져옴
16         Item item = this.shopService.getItemById(itemId);
17
18         // 모델 생성
19         Map<String, Object> model = new HashMap<String, Object>();
20         model.put("item", item);
21
22         // 반환값인 ModelAndView 인스턴스를 생성
23         ModelAndView modelAndView = new ModelAndView();
24         modelAndView.setViewName("itemDetail");
25         modelAndView.addObject(model);
26
27         return modelAndView;
28     }
29 }
30 }
```

ItemId로 객체정보를 찾아서 해당 정보를 item에 넣고 itemDetail View 출력

ItemListController

```
1 package com.spring.input.controller;
2
3 import java.util.ArrayList;
4
5 public class ItemListController implements Controller {
6     private Shop shopService;
7
8     public void setShopService(Shop shopService) {
9         this.shopService = shopService;
10     }
11
12     public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response) throws Exception {
13
14         ArrayList<Item> itemList = this.shopService.getItemList();
15
16         Map<String, Object> model = new HashMap<String, Object>();
17         model.put("itemList", itemList);
18
19         ModelAndView modelAndView = new ModelAndView();
20         modelAndView.setViewName("itemList");
21         modelAndView.addObject(model);
22
23         return modelAndView;
24     }
25 }
```

ArrayList<Item>로 itemList에 입력된 정보들을 모두 넣고 이것을 ItemList View 로 출력

ItemDao

```
1 package com.spring.input.dao;
2
3 import java.util.ArrayList;
4
5 public interface ItemDao {
6
7     void create(Item item);
8     void delete(Item item);
9
10     ArrayList<Item> findAll();
11
12     Item findById(String itemId);
13     Item findByName(String itemName);
14 }
```

Interface 클래스로 Item에 관련된 함수이름들을 정의

ItemDaoImpl

```
7 public class ItemDaoImpl implements ItemDao {
8
9     public ArrayList<Item> itemList = new ArrayList<Item>();
10
11     public void create(Item item) {
12         itemList.add(item);
13     }
14
15     public void delete(Item item) {
16         itemList.clear();
17     }
18
19     public Item getItem(String itemId) {
20         for(int i=0; i<this.itemList.size(); i++) {
21             if (itemList.get(i).getItemId() == itemId)
22                 return itemList.get(i);
23         }
24         return null;
25     }
26
27     @Override
28     public ArrayList<Item> findAll() {
29         return itemList;
30     }
31
32     @Override
33     public Item findById(String itemId) {
34         Item item = new Item();
35         for (int i = 0; i < itemList.size(); i++) {
36             if (itemList.get(i).getItemId().equals(itemId)) {
37                 item = itemList.get(i);
38                 break;
39             }
40         }
41         return item;
42     }
43
44     public Item findByName(String itemName) {
45         Item item = new Item();
46         for (int i = 0; i < itemList.size(); i++) {
47             if (itemList.get(i).getItemName().equals(itemName)) {
48                 item = itemList.get(i);
49                 break;
50             }
51         }
52         return item;
53     }
54 }
```

ItemDao의 구현 클래스로 Item에 관련된 함수body들을 정의

- create : 상품 추가
- delete : 상품 삭제
- findAll : 모두 찾기
- findById : 상품ID로 상품찾기
- findByName : 상품명으로 상품찾기

UserDao

```
1 package com.spring.input.dao;
2
3 import java.util.ArrayList;
4
5 public interface UserDao {
6
7     void create(User user);
8
9     ArrayList<User> findAll();
10
11     User findById(String userId);
12 }
13
14 }
```

Interface 클래스로 User에 관련된 함수이름들을 정의

UserDaoImpl

```
1 package com.spring.input.dao;
2
3 import java.util.ArrayList;
4
5 public class UserDaoImpl implements UserDao {
6
7     public ArrayList<User> userList = new ArrayList<User>();
8
9     public void create(User user) {
10         user.setUserId(user.getUserId());
11         userList.add(user);
12     }
13
14     public ArrayList<User> findAll() {
15         return userList;
16     }
17
18     public User findById(String userId) {
19         User user = new User();
20         for (int i = 0; i < userList.size(); i++) {
21             if (userList.get(i).getUserId().equals(userId)) {
22                 user = userList.get(i);
23                 break;
24             }
25         }
26         return user;
27     }
28 }
29
30
31
32
```

UserDao의 구현 클래스로 User에 관련된 함수body들을 정의

- create : 유저 추가
- findAll : 모두 찾기
- findById : 유저ID로 유저찾기

Item

```
1 package com.spring.input.logic;
2
3 import java.io.Serializable;
4
5 public class Item implements Serializable{
6
7     private static final long serialVersionUID = 1L;
8
9     private String itemId; //아이템 id
10    private String itemClass; // 아이템 종류
11    private String itemName; // 아이템 이름
12    private String description; // 설명
13    private String frequency; // 사용 주기
14    private String term; // 사용 기간
15    private String money; // 비용금액
16
17    public String getMoney() {return money;}
18    public void setMoney(String money) {this.money = money;}
19    public String getDescription() {return this.description;}
20    public void setDescription(String description) {this.description = description;}
21    public String getItemId() {return this.itemId;}
22    public void setItemId(String itemId) {this.itemId = itemId;}
23    public String getItemName() {return this.itemName;}
24    public void setItemName(String itemName) {this.itemName = itemName;}
25    public String getFrequency() {return frequency;}
26    public void setFrequency(String frequency) {this.frequency = frequency;}
27    public String getTerm() {return term;}
28    public void setTerm(String term) {this.term = term;}
29    public String getItemClass() {return itemClass;}
30    public void setItemClass(String itemClass) {this.itemClass = itemClass;}
31 }

```

Item이 가지는 정보들

ItemCatalog

```
1 package com.spring.input.logic;
2
3 import java.util.ArrayList;
4
5 public interface ItemCatalog {
6
7     void entryItem (Item item);
8     Item getItemById (String itemId);
9     ArrayList<Item> getItemList();
10    Item getItemByItemName(String itemName);
11 }

```

Interface 클래스로 ItemDao에 관련된 함수이름들을 정의

ItemCatalogImpl

```
1 package com.spring.input.logic;
2
3 import java.util.ArrayList;
4
5 public class ItemCatalogImpl implements ItemCatalog {
6
7     private ItemDao itemDao;
8
9     public void setItemDao(ItemDao itemDao) {
10         this.itemDao = itemDao;
11     }
12
13     public void entryItem(Item item) {
14         this.itemDao.create(item);
15     }
16
17     public void deleteItem(Item item) {
18         this.itemDao.delete(item);
19     }
20
21     public ArrayList<Item> getItemList() {
22         return this.itemDao.findAll();
23     }
24
25     public Item getItemById(String itemId) {
26         return this.itemDao.findById(itemId);
27     }
28
29     public Item getItemByName(String itemName) {
30         return this.itemDao.findByName(itemName);
31     }
32
33 }
34
35 }
```

ItemCatalog의 구현 클래스로 ItemDao에 관련된 함수body들을 정의

- setItemDao : itemDao 연결
- entryItem : 입력받은 상품 itemDao에 넘겨줘서 추가
- deleteItem : 삭제할 상품 itemDao에 넘겨줘서 삭제
- getItemList : 상품리스트 itemDao에서 모두 가져오기
- getItemById : itemDao에서 itemId로 찾아서 가져오기
- getItemByName : itemDao에서 itemName으로 찾아서 가져오기

User

```
1 package com.spring.input.logic;
2
3 import java.io.Serializable;
4
5 public class User implements Serializable {
6
7     private static final long serialVersionUID = 1L;
8
9     private String userId;
10    private String userName;
11    private String age;
12    private String ssn;
13    private String address;
14    private String email;
15    private String job;
16
17    public String getAddress() {return this.address;}
18    public void setAddress(String address) {this.address = address;}
19    public String getEmail() {return this.email;}
20    public void setEmail(String email) {this.email = email;}
21    public String getJob() {return this.job;}
22    public void setJob(String job) {this.job = job;}
23    public String getAge() {return this.age;}
24    public void setAge(String age) {this.age = age;}
25    public String getSsn() {return this.ssn;}
26    public void setSsn(String ssn) {this.ssn = ssn;}
27    public String getUserId() {return this.userId;}
28    public void setUserId(String userId) {this.userId = userId;}
29    public String getUserName() {return this.userName;}
30    public void setUserName(String userName) {this.userName = userName;}
31 }
```

User가 가지는 정보들

UserCatalog

```
1 package com.spring.input.logic;
2
3 import java.util.ArrayList;
4
5 public interface UserCatalog {
6
7     void entryUser (User user);
8     User getUserById (String userId);
9     ArrayList<User> getUserList();
10
11 }
```

Interface 클래스로 UserDao에 관련된 함수이름들을 정의

UserCatalogImpl

```
1 package com.spring.input.logic;
2
3 import java.util.ArrayList;
4
5 public class UserCatalogImpl implements UserCatalog {
6
7     private UserDao userDao;
8
9     public void setUserDao(UserDao userDao) {
10         this.userDao = userDao;
11     }
12
13     public void entryUser(User user) {
14         this.userDao.create(user);
15     }
16
17     public ArrayList<User> getUserList() {
18         return this.userDao.findAll();
19     }
20
21     public User getUserById(String userId) {
22         return this.userDao.findById(userId);
23     }
24 }
25
26 }
```

UserCatalog의 구현 클래스로 UserDao에 관련된 함수body들을 정의

- setUserDao : userDao 연결
- entryUser : 입력받은 유저 userDao에 넘겨줘서 추가
- getUserList : 유저리스트 userDao에서 모두 가져오기
- getUserById : userDao에서 userId로 찾아서 가져오기

Shop

```
1 package com.spring.input.logic;
2
3 import java.util.ArrayList;
4
5 public interface Shop {
6
7     void entryItem(Item item);
8     void entryUser(User user);
9
10     ArrayList<Item> getItemList();
11     ArrayList<User> getUserList();
12
13     Item getItemById(String itemId);
14     Item getItemByName(String itemName);
15     User getUserById(String userId);
16 }
```

Interface 클래스로 ItemCatalog, UserCatalog에 관련된 함수이름들을 정의

ShopImpl

```
1 package com.spring.input.logic;
2
3 import java.util.ArrayList;
4
5 public class ShopImpl implements Shop {
6
7     private ItemCatalog itemCatalog;
8     private UserCatalog userCatalog;
9
10     //item
11
12     public void setItemCatalog(ItemCatalog itemCatalog) {
13         this.itemCatalog = itemCatalog;
14     }
15     public void entryItem(Item item) {
16         this.itemCatalog.entryItem(item);
17     }
18     public ArrayList<Item> getItemList() {
19         return this.itemCatalog.getItemList();
20     }
21     public Item getItemById(String itemId) {
22         return this.itemCatalog.getItemById(itemId);
23     }
24     public Item getItemByName(String itemName) {
25         return this.itemCatalog.getItemByName(itemName);
26     }
27
28     //user
29
30     public void setUserCatalog(UserCatalog userCatalog) {
31         this.userCatalog = userCatalog;
32     }
33     public void entryUser(User user) {
34         this.userCatalog.entryUser(user);
35     }
36     public ArrayList<User> getUserList() {
37         return this.userCatalog.getUserList();
38     }
39     public User getUserById(String userId) {
40         return this.userCatalog.getUserById(userId);
41     }
42 }
```

Shop의 구현 클래스로 ItemCatalog, UserCatalog에 관련된 함수body들을 정의

- setItemCatalog : itemCatalog 연결
- entryItem : 입력받은 상품 itemCatalog에 넘겨줘서 추가
- deleteItem : 삭제할 상품 itemCatalog에 넘겨줘서 삭제
- getItemList : 상품리스트 itemCatalog에서 모두 가져오기
- getItemById : itemCatalog에서 itemId로 찾아서 가져오기
- getItemByName : itemCatalog에서 itemName으로 찾아서 가져오기
- setUserCatalog : userCatalog 연결
- entryUser : 입력받은 유저 userCatalog에 넘겨줘서 추가
- getUserList : 유저리스트 userCatalog에서 모두 가져오기
- getUserById : userCatalog에서 userId로 찾아서 가져오기

ItemEntryValidator

```

1 package com.spring.input.utils;
2
3 import com.spring.input.logic.Item;
4
5 public class ItemEntryValidator implements Validator {
6
7     public boolean supports(Class<?> clazz) {
8         return Item.class.isAssignableFrom(clazz);
9     }
10
11     public void validate(Object command, Errors errors) {
12         Item user = (Item) command;
13
14         if (StringUtils.hasLength(user.getItemId())) {
15             errors.rejectValue("itemId", "error.required");
16         }
17
18         if (StringUtils.hasLength(user.getItemName())) {
19             errors.rejectValue("itemName", "error.required");
20         }
21
22         if (StringUtils.hasText(user.getDescription())) {
23             errors.rejectValue("description", "error.required");
24         }
25
26         if (errors.hasErrors()) {
27             errors.reject("error.input.user");
28         }
29     }
30 }

```

에러 체크 (에러 발생시 messages properties에서 호출하여 출력)

UserEntryValidator

```

1 package com.spring.input.utils;
2
3 import com.spring.input.logic.User;
4
5 public class UserEntryValidator implements Validator {
6
7     public boolean supports(Class<?> clazz) {
8         return User.class.isAssignableFrom(clazz);
9     }
10
11     public void validate(Object command, Errors errors) {
12         User user = (User) command;
13
14         if (StringUtils.hasLength(user.getUserId())) {
15             errors.rejectValue("userId", "error.required");
16         }
17
18         if (StringUtils.hasLength(user.getAge())) {
19             errors.rejectValue("age", "error.required");
20         }
21
22         if (StringUtils.hasLength(user.getUserName())) {
23             errors.rejectValue("userName", "error.required");
24         }
25
26         if (StringUtils.hasText(user.getSsn())) {
27             errors.rejectValue("ssn", "error.required");
28         }
29
30         if (StringUtils.hasText(user.getAddress())) {
31             errors.rejectValue("address", "error.required");
32         }
33
34         if (StringUtils.hasText(user.getEmail())) {
35             errors.rejectValue("email", "error.required");
36         }
37
38         if (errors.hasErrors()) {
39             errors.reject("error.input.user");
40         }
41     }
42 }

```

에러 체크 (에러 발생시 messages properties에서 호출하여 출력)

3) 실행결과

보험 설계하기

1. 보험정보 입력 후 등록

← → ↻ localhost:8181/inputTest/itemEntry.html ☆

NAVER 학교 SW개발 시스템과 프로그래밍 Github 기타 분산프로그래밍

보험 설계 화면

보험ID

보험종류

보험이름

설명

납입주기

납입기간

보상금액

2. 보험 생성 완료

← → ↻ localhost:8181/inputTest/itemEntry.html ☆

NAVER 학교 SW개발 시스템과 프로그래밍 Github 기타 분산프로그래밍

보험 생성 완료 화면

보험 생성이 완료되었습니다.

상품ID	암보험
상품ID	123
보험명	생명사랑암보험
납입주기	1
납입기간	10
보상금액	1억
설명	이 보험은 보상금액 1억의 10년 만기 암보험입니다.

보험 확정하기

- 이전에 생성된 여러 보험상품들 중 확정 거절할 보험 선택하여 삭제 (삭제하지 않으면 확정된 상품)

localhost:8181/inputTest/itemList.html

NAVER 학교 SW개발 시스템과 프로그래밍 Github 기타 본산프로그래밍

보험 확정 화면

확정 거절할 상품을 체크하여 삭제해주세요

상품명을 누르면 상세화면으로 이동하며, 보험가입을 신청할 수 있습니다

선택	보험ID	보험종류	보험명	납입주기	납입기간	보상금액
<input type="checkbox"/>	123	암보험	생명사할암보험	1달	10년	1억원
<input type="checkbox"/>	10	종신보험	life보험	1달	25년	1억원
<input type="checkbox"/>	5	연금보험	육둔마련보험	1달	10년	5천만원
<input type="checkbox"/>	77	실비보험	실속check보험	2달	5년	2백만원

확정 거절

← 입력화면으로

- 선택하여 확정 거절

localhost:8181/inputTest/itemList.html

NAVER 학교 SW개발 시스템과 프로그래밍 Github 기타 본산프로그래밍

보험 확정 화면

확정 거절할 상품을 체크하여 삭제해주세요

상품명을 누르면 상세화면으로 이동하며, 보험가입을 신청할 수 있습니다

선택	보험ID	보험종류	보험명	납입주기	납입기간	보상금액
<input type="checkbox"/>	123	암보험	생명사할암보험	1달	10년	1억원
<input type="checkbox"/>	10	종신보험	life보험	1달	25년	1억원
<input checked="" type="checkbox"/>	5	연금보험	육둔마련보험	1달	10년	5천만원
<input checked="" type="checkbox"/>	77	실비보험	실속check보험	2달	5년	2백만원

확정 거절

← 입력화면으로

localhost:8181/inputTest/itemList.html

NAVER 학교 SW개발 시스템과 프로그래밍 Github 기타 본산프로그래밍

보험 확정 화면

확정 거절할 상품을 체크하여 삭제해주세요

상품명을 누르면 상세화면으로 이동하며, 보험가입을 신청할 수 있습니다

선택	보험ID	보험종류	보험명	납입주기	납입기간	보상금액
<input type="checkbox"/>	123	암보험	생명사할암보험	1달	10년	1억원
<input type="checkbox"/>	10	종신보험	life보험	1달	25년	1억원

확정 거절

← 입력화면으로

보험 가입하기

5. 가입을 원하는 보험의 상세정보 클릭 후, 가입하기 클릭

localhost:8181/inputTest/itemDetail.html?itemId=10

NAVER 학교 SW개발 시스템과 프로그래밍 Github 기타 분산프로그래밍

보험 상세 화면

가입하시려면 가입하기 버튼을 눌러주세요.

보험ID	보험종류	이름	납입주기	납입기간	보상금액	설명
10	종신보험	Life보험	1	25	1억	이 보험은 종신보험입니다.

목록 보기 | 상세화면으로 | 가입 하기

6. 가입자 정보를 입력 후 등록

localhost:8181/inputTest/userEntry.html

NAVER 학교 SW개발 시스템과 프로그래밍 Github 기타 분산프로그래밍

보험 가입 화면

가입자의 정보를 입력하세요

유저ID: 16

이름: 박준현

나이: 23

주민등록번호: 960809-1234567

주소: 서울시 서대문구

E-MAIL: hi@gmail.com

직업: 학생

등록 | 리셋

목록 보기 | 상세화면으로

7. 가입 완료

localhost:8181/inputTest/userEntry.html

NAVER 학교 SW개발 시스템과 프로그래밍 Github 기타 분산프로그래밍

보험 가입신청 완료 화면

보험 가입신청이 완료되었습니다.
UW가 완료되면 보험 가입 여부를 알려드립니다.

유저ID	16
이름	박준현
나이	23
주민등록번호	960809-1234567
주소	서울시 서대문구
E-MAIL	hi@gmail.com
직업	학생

목록 보기