< 분산 프로그래밍 >

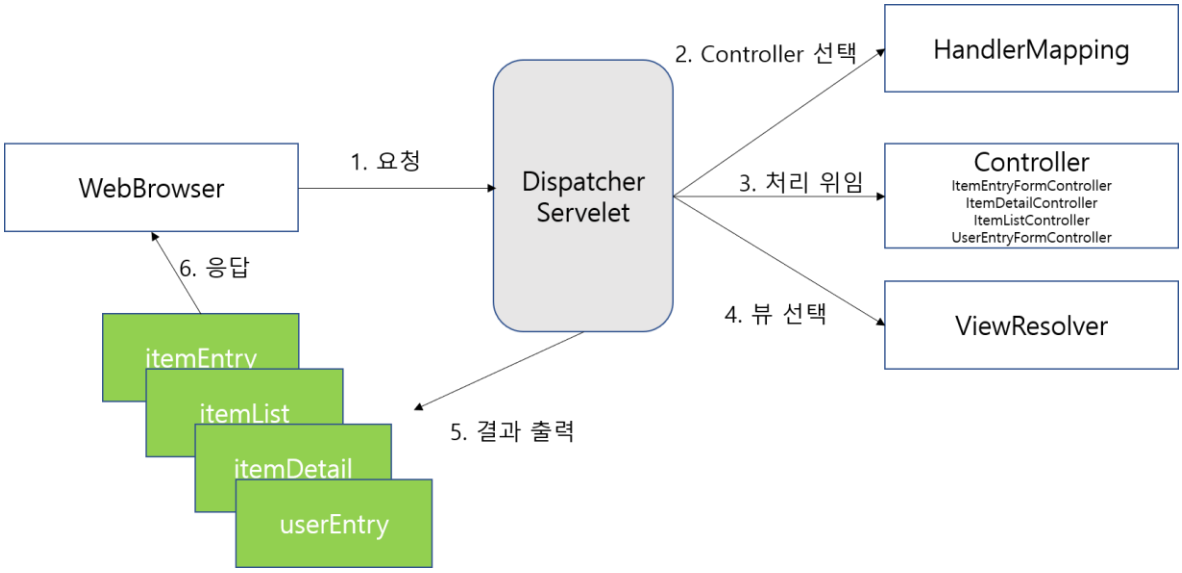
기말 REPORT

목차

과목명	분산프로그래밍
교수명	김정호
학과	융합소프트웨어학부
학번	60161606
이름	박준현

1. 설계

1) SpringMVC



- 클라이언트에서 URL로 접근하여 DispatcherServelet에 정보를 요청
- DispatcherServelet은 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

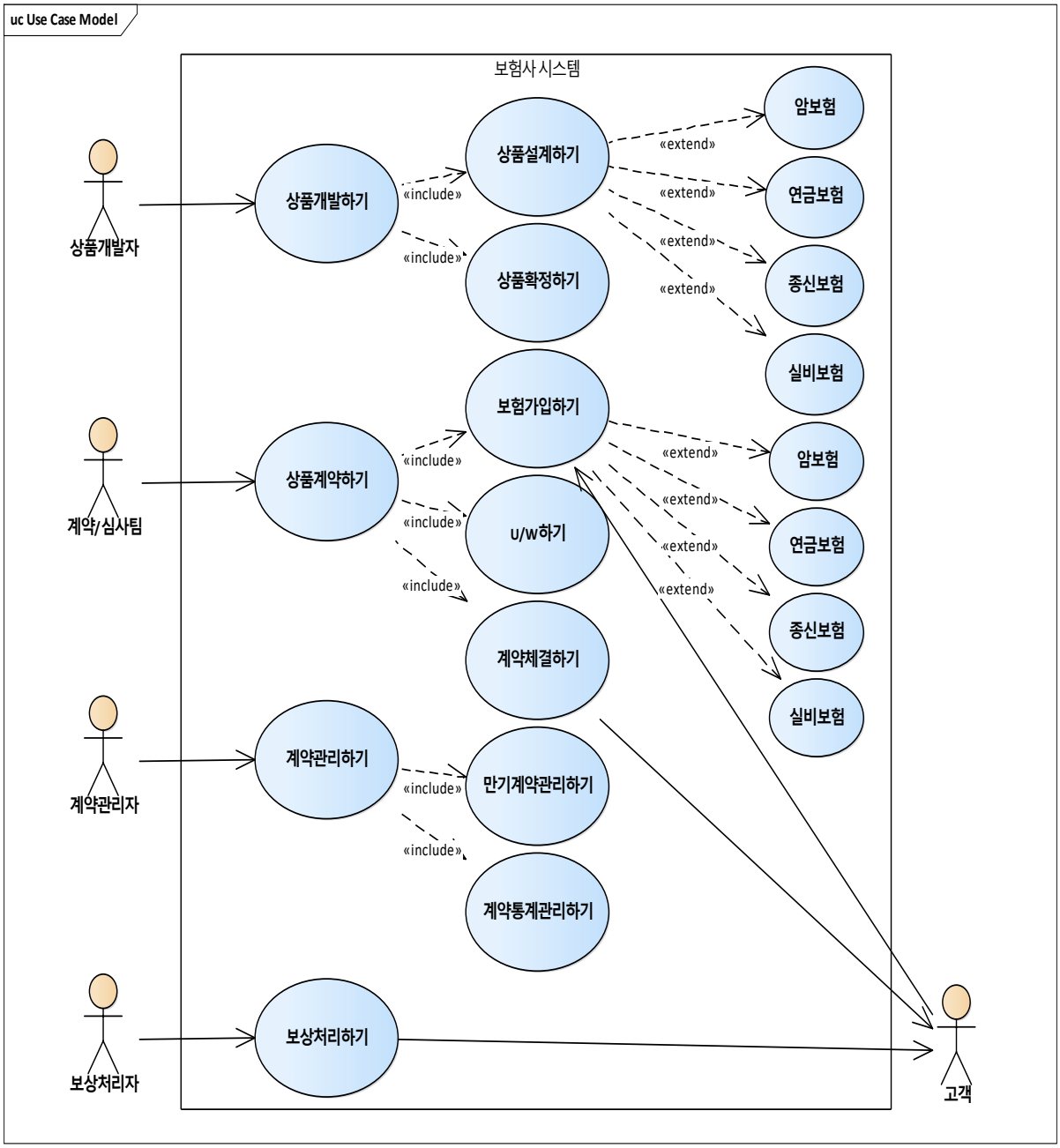
<Controller별 역할>

- ItemEntryFromController : 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

Servelet-context.xml

```
<!-- HandlerMapping -->
<bean id="handlerMapping"
    class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <value>
            /itemEntry.html=itemEntryFormController
            /itemList.html=itemListController
            /itemDetail.html=itemDetailController
            /userEntry.html=userEntryFormController
        </value>
    </property>
</bean>
```

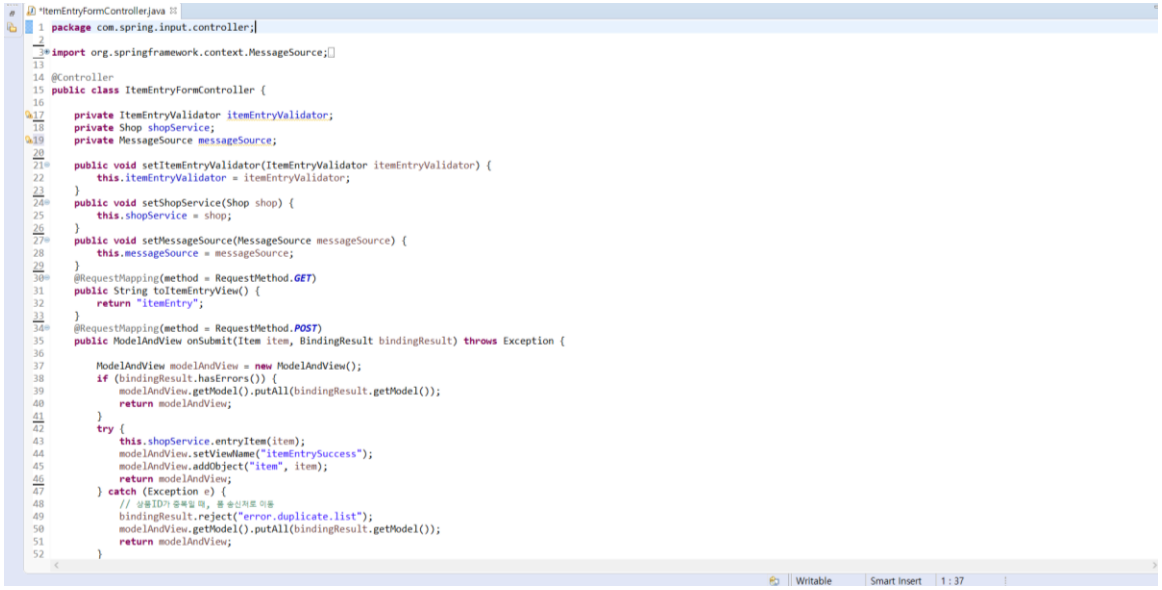
HandlerMapping을 통해 View별 Controller 배정

```
<!-- Controller -->
<bean id="itemEntryFormController" class="com.spring.input.controller.ItemEntryFormController"
    p:shopService-ref="shopService"
    p:itemEntryValidator-ref="itemEntryValidator"
    p:messageSource-ref="messageSource">
</bean>
<bean id="userEntryFormController" class="com.spring.input.controller.UserEntryFormController"
    p:shopService-ref="shopService"
    p:userEntryValidator-ref="userEntryValidator"
    p:messageSource-ref="messageSource">
</bean>
<bean id="itemListController" class="com.spring.input.controller.ItemListController"
    p:shopService-ref="shopService">
</bean>
<bean id="itemDetailController" class="com.spring.input.controller.ItemDetailController"
    p:shopService-ref="shopService">
</bean>
<bean id="shopService" class="com.spring.input.logic.ShopImpl"
    p:itemCatalog-ref="itemCatalog"
    p:userCatalog-ref="userCatalog">
</bean>
<bean id="itemEntryValidator" class="com.spring.input.utils.ItemEntryValidator"/>
<bean id="userEntryValidator" class="com.spring.input.utils.UserEntryValidator"/>
<bean id="messageSource"
 class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
    <property name="basenames">
            <list>
                <value>resources/messages</value>
            </list>
    </property>
</bean>
<bean class="com.spring.input.logic.UserCatalogImpl" id="userCatalog"
    p:userDao-ref="userDao">
</bean>
<bean class="com.spring.input.dao.UserDaoImpl" id="userDao">
</bean>
<bean class="com.spring.input.logic.ItemCatalogImpl" id="itemCatalog"
    p:itemDao-ref="itemDao">
</bean>
<bean class="com.spring.input.dao.ItemDaoImpl" id="itemDao">
</bean>
```

각 class의 bean생성

```
<!-- ViewResolver -->
<bean id="internalResourceViewResolver"
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass">
        <value>org.springframework.web.servlet.view.JstlView</value>
    </property>
    <property name="prefix">
        <value>WEB-INF/views/</value>
    </property>
    <property name="suffix">
        <value>.jsp</value>
    </property>
</bean>
</beans>
```

ItemEntryFormController

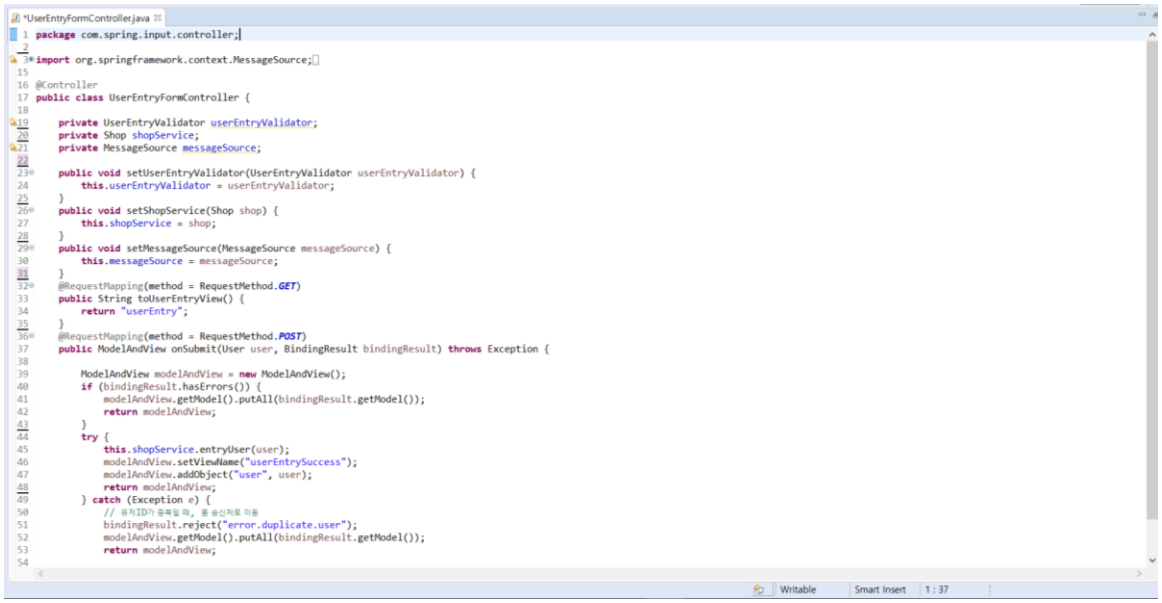


```java
package com.spring.input.controller;

import org.springframework.context.MessageSource;

@Controller
public class ItemEntryFormController {

    private ItemEntryValidator itemEntryValidator;
    private Shop shopService;
    private MessageSource messageSource;

    public void setItemEntryValidator(ItemEntryValidator itemEntryValidator) {
        this.itemEntryValidator = itemEntryValidator;
    }
    public void setShopService(Shop shop) {
        this.shopService = shop;
    }
    public void setMessageSource(MessageSource messageSource) {
        this.messageSource = messageSource;
    }
    @RequestMapping(method = RequestMethod.GET)
    public String toItemEntryView() {
        return "itemEntry";
    }
    @RequestMapping(method = RequestMethod.POST)
    public ModelAndView onSubmit(Item item, BindingResult bindingResult) throws Exception {

        ModelAndView modelAndView = new ModelAndView();
        if (bindingResult.hasErrors()) {
            modelAndView.getModel().putAll(bindingResult.getModel());
            return modelAndView;
        }
        try {
            this.shopService.entryItem(item);
            modelAndView.setViewName("itemEntrySuccess");
            modelAndView.addObject("item", item);
            return modelAndView;
        } catch (Exception e) {
            // 상품ID가 중복될 때, 롤 송신처로 이동
            bindingResult.reject("error.duplicate.list");
            modelAndView.getModel().putAll(bindingResult.getModel());
            return modelAndView;
        }
```

정보입력시 Validator가 입력정보 판독

입력이 성공적으로 완료되면 itemEntrySuccess View 띄움

UserEntryFormController



```java
package com.spring.input.controller;

import org.springframework.context.MessageSource;

@Controller
public class UserEntryFormController {

    private UserEntryValidator userEntryValidator;
    private Shop shopService;
    private MessageSource messageSource;

    public void setUserEntryValidator(UserEntryValidator userEntryValidator) {
        this.userEntryValidator = userEntryValidator;
    }
    public void setShopService(Shop shop) {
        this.shopService = shop;
    }
    public void setMessageSource(MessageSource messageSource) {
        this.messageSource = messageSource;
    }
    @RequestMapping(method = RequestMethod.GET)
    public String toUserEntryView() {
        return "userEntry";
    }
    @RequestMapping(method = RequestMethod.POST)
    public ModelAndView onSubmit(User user, BindingResult bindingResult) throws Exception {

        ModelAndView modelAndView = new ModelAndView();
        if (bindingResult.hasErrors()) {
            modelAndView.getModel().putAll(bindingResult.getModel());
            return modelAndView;
        }
        try {
            this.shopService.entryUser(user);
            modelAndView.setViewName("userEntrySuccess");
            modelAndView.addObject("user", user);
            return modelAndView;
        } catch (Exception e) {
            // 유저ID가 중복될 때, 롤 송신처로 이동
            bindingResult.reject("error.duplicate.user");
            modelAndView.getModel().putAll(bindingResult.getModel());
            return modelAndView;
```

정보입력시 Validator가 입력정보 판독

입력이 성공적으로 완료되면 userEntrySuccess View 띄움

ItemDetailController

```java
package com.spring.input.controller;

import java.util.HashMap;

@Controller
public class ItemDetailController {

    private Shop shopService;

    public void setShopService(Shop shop) {
        this.shopService = shop;
    }

    @RequestMapping
    public ModelAndView detailItem(String itemId){
        // 선택된 상품ID에서 상품 정보를 취득
        Item item = this.shopService.getItemByItemId(itemId);

        // 모델 생성
        Map<String, Object> model = new HashMap<String, Object>();
        model.put("item", item);

        // 반환값인 ModelAndView 인스턴스를 생성
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("itemDetail");
        modelAndView.addAllObjects(model);

        return modelAndView;

    }
}
```

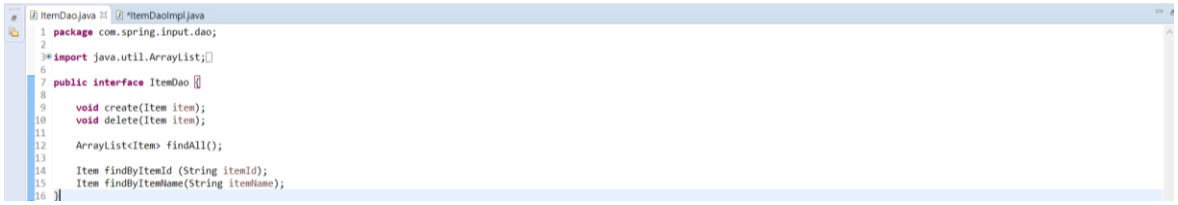
ItemId로 객체정보를 찾아서 해당 정보를 item에 넣고 itemDetail View 출력

ItemListController

```java
package com.spring.input.controller;

import java.util.ArrayList;

public class ItemListController implements Controller {

    private Shop shopService;

    public void setShopService(Shop shopService) {
        this.shopService = shopService;
    }

    public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response) throws Exception {


        ArrayList<Item> itemList = this.shopService.getItemList();

        Map<String, Object> model = new HashMap<String, Object>();
        model.put("itemList", itemList);

        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("itemList");
        modelAndView.addAllObjects(model);

        return modelAndView;
    }
}
```

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

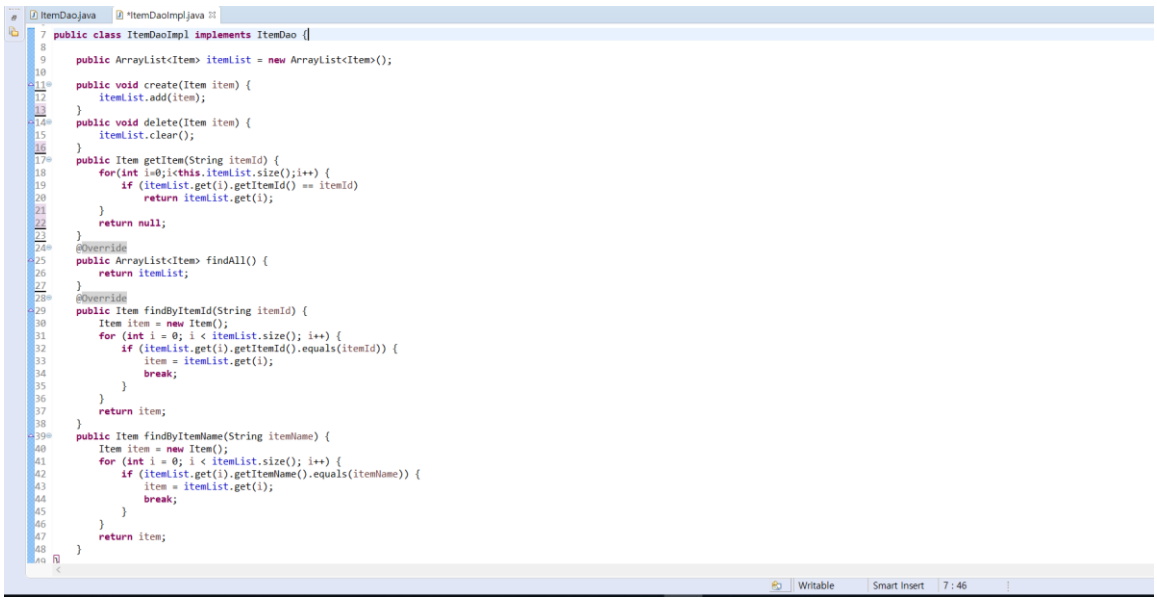
ItemDao



```java
package com.spring.input.dao;

import java.util.ArrayList;

public interface ItemDao {

    void create(Item item);
    void delete(Item item);

    ArrayList<Item> findAll();

    Item findByItemId (String itemId);
    Item findByItemName(String itemName);
}
```

Interface 클래스로 Item에 관련된 함수이름들을 정의

ItemDaoImpl

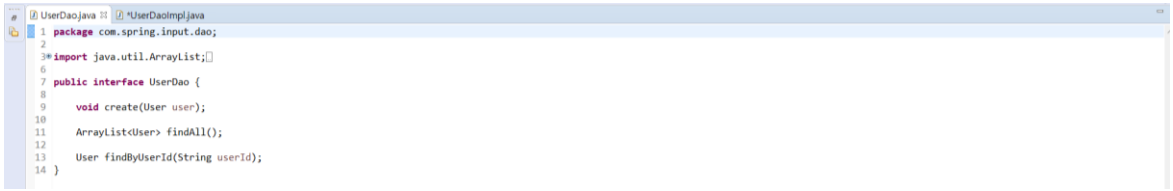


```java
public class ItemDaoImpl implements ItemDao {

    public ArrayList<Item> itemList = new ArrayList<Item>();

    public void create(Item item) {
        itemList.add(item);
    }
    public void delete(Item item) {
        itemList.clear();
    }
    public Item getItem(String itemId) {
        for(int i=0;i<this.itemList.size();i++) {
            if (itemList.get(i).getItemId() == itemId)
                return itemList.get(i);
        }
        return null;
    }
    @Override
    public ArrayList<Item> findAll() {
        return itemList;
    }
    @Override
    public Item findByItemId(String itemId) {
        Item item = new Item();
        for (int i = 0; i < itemList.size(); i++) {
            if (itemList.get(i).getItemId().equals(itemId)) {
                item = itemList.get(i);
                break;
            }
        }
        return item;
    }
    public Item findByItemName(String itemName) {
        Item item = new Item();
        for (int i = 0; i < itemList.size(); i++) {
            if (itemList.get(i).getItemName().equals(itemName)) {
                item = itemList.get(i);
                break;
            }
        }
        return item;
    }
```

ItemDao의 구현 클래스로 Item에 관련된 함수body들을 정의

- create : 상품 추가
- delete : 상품 삭제
- findAll : 모두 찾기
- findByItemId : 상품ID로 상품찾기
- findByItemName : 상품이름으로 상품찾기

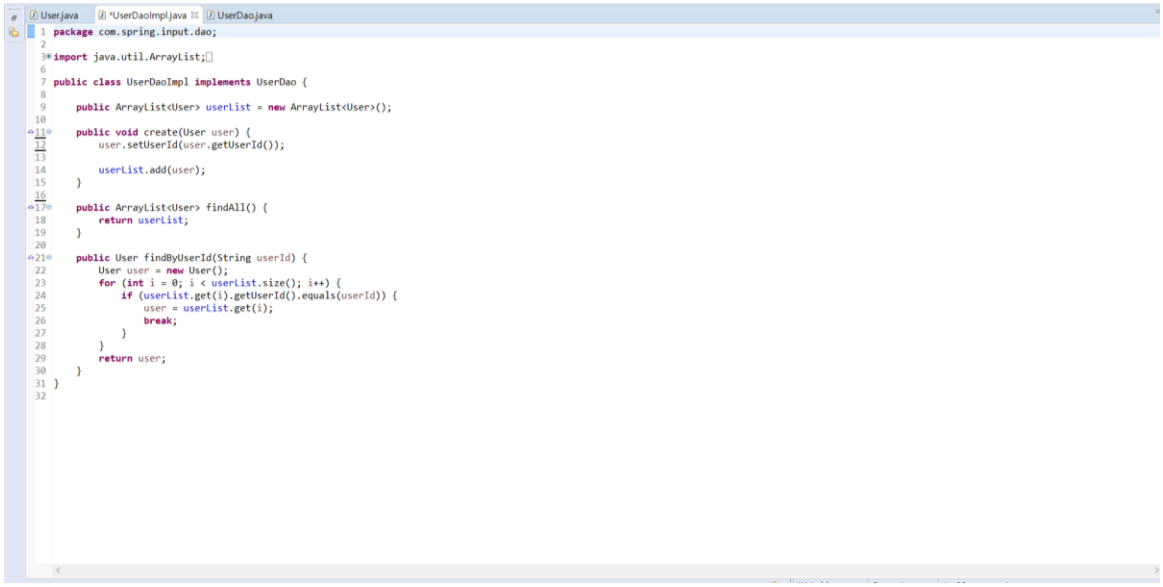
UserDao



```java
package com.spring.input.dao;

import java.util.ArrayList;

public interface UserDao {

    void create(User user);

    ArrayList<User> findAll();

    User findByUserId(String userId);
}
```

Interface 클래스로 User에 관련된 함수이름들을 정의

UserDaoImpl



```java
package com.spring.input.dao;

import java.util.ArrayList;

public class UserDaoImpl implements UserDao {

    public ArrayList<User> userList = new ArrayList<User>();

    public void create(User user) {
        user.setUserId(user.getUserId());

        userList.add(user);
    }

    public ArrayList<User> findAll() {
        return userList;
    }

    public User findByUserId(String userId) {
        User user = new User();
        for (int i = 0; i < userList.size(); i++) {
            if (userList.get(i).getUserId().equals(userId)) {
                user = userList.get(i);
                break;
            }
        }
        return user;
    }
}
```

UserDao의 구현 클래스로 User에 관련된 함수body들을 정의

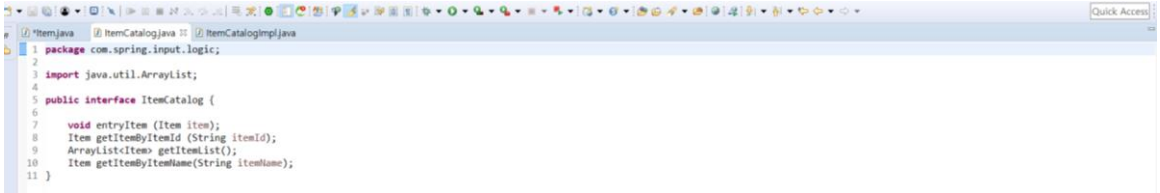
- create : 유저 추가
- findAll : 모두 찾기
- findByUserId : 유저ID로 유저찾기

Item

```java
package com.spring.input.logic;

import java.io.Serializable;

public class Item implements Serializable{

    private static final long serialVersionUID = 1L;

    private String itemId; //보험 id
    private String itemClass; // 보험 종류
    private String itemName; // 보험 이름
    private String description; // 보험 설명
    private String frequency; // 납입 주기
    private String term; // 납입 기간
    private String money; // 보상금액

    public String getMoney() {return money;}
    public void setMoney(String money) {this.money = money; }
    public String getDescription() {return this.description;}
    public void setDescription(String description) {this.description = description;}
    public String getItemId() {return this.itemId;}
    public void setItemId(String itemId) {this.itemId = itemId;}
    public String getItemName() {return this.itemName;}
    public void setItemName(String itemName) {this.itemName = itemName;}
    public String getFrequency() {return frequency;}
    public void setFrequency(String frequency) {this.frequency = frequency;}
    public String getTerm() {return term;}
    public void setTerm(String term) {this.term = term;}
    public String getItemClass() {return itemClass;}
    public void setItemClass(String itemClass) {this.itemClass = itemClass;}
}
```

Item이 가지는 정보들

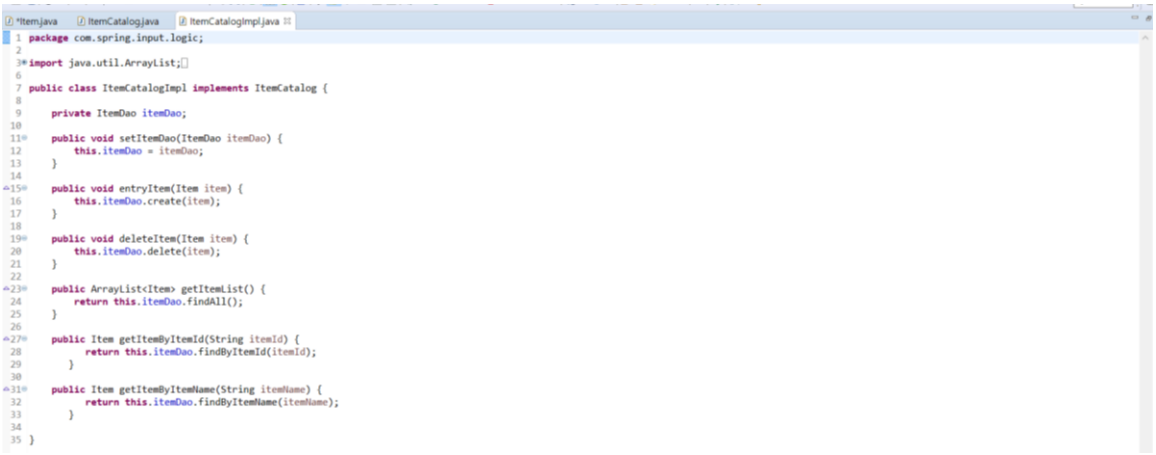
ItemCatalog



```java
package com.spring.input.logic;

import java.util.ArrayList;

public interface ItemCatalog {

    void entryItem (Item item);
    Item getItemByItemId (String itemId);
    ArrayList<Item> getItemList();
    Item getItemByItemName(String itemName);
}
```

Interface 클래스로 ItemDao에 관련된 함수이름들을 정의

ItemCatalogImpl

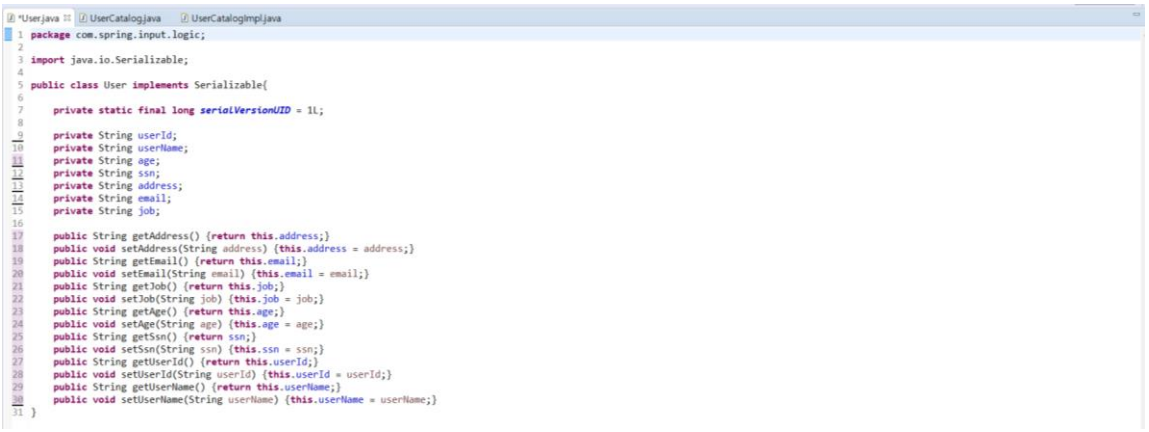


```java
package com.spring.input.logic;

import java.util.ArrayList;

public class ItemCatalogImpl implements ItemCatalog {

    private ItemDao itemDao;

    public void setItemDao(ItemDao itemDao) {
        this.itemDao = itemDao;
    }

    public void entryItem(Item item) {
        this.itemDao.create(item);
    }

    public void deleteItem(Item item) {
        this.itemDao.delete(item);
    }

    public ArrayList<Item> getItemList() {
        return this.itemDao.findAll();
    }

    public Item getItemByItemId(String itemId) {
        return this.itemDao.findByItemId(itemId);
    }

    public Item getItemByItemName(String itemName) {
        return this.itemDao.findByItemName(itemName);
    }

}
```

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
- getItemByItemId : itemDao에서 itemId로 찾아서 가져오기
- getItemByItemName : itemDao에서 itemName으로 찾아서 가져오기

User



```java
package com.spring.input.logic;

import java.io.Serializable;

public class User implements Serializable{

    private static final long serialVersionUID = 1L;

    private String userId;
    private String userName;
    private String age;
    private String ssn;
    private String address;
    private String email;
    private String job;

    public String getAddress() {return this.address;}
    public void setAddress(String address) {this.address = address;}
    public String getEmail() {return this.email;}
    public void setEmail(String email) {this.email = email;}
    public String getJob() {return this.job;}
    public void setJob(String job) {this.job = job;}
    public String getAge() {return this.age;}
    public void setAge(String age) {this.age = age;}
    public String getSsn() {return ssn;}
    public void setSsn(String ssn) {this.ssn = ssn;}
    public String getUserId() {return this.userId;}
    public void setUserId(String userId) {this.userId = userId;}
    public String getUserName() {return this.userName;}
    public void setUserName(String userName) {this.userName = userName;}
}
```

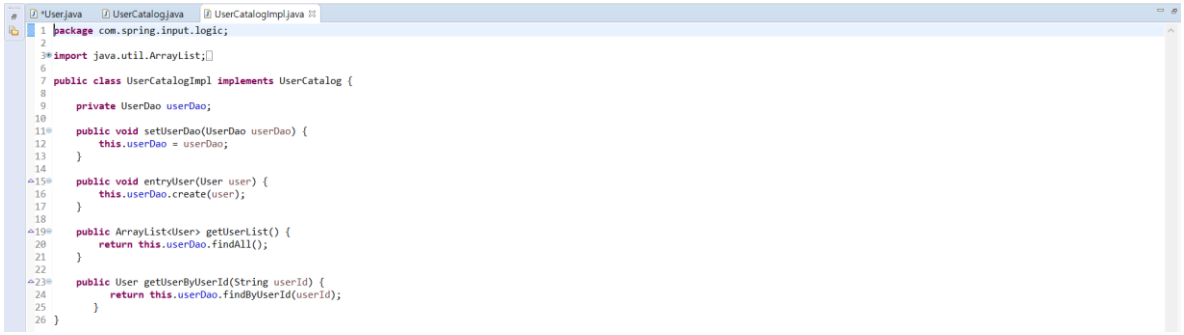
User가 가지는 정보들

UserCatalog

```java
package com.spring.input.logic;

import java.util.ArrayList;

public interface UserCatalog {

    void entryUser (User user);
    User getUserByUserId (String userId);
    ArrayList<User> getUserList();

}
```

Interface 클래스로 UserDao에 관련된 함수이름들을 정의

UserCatalogImpl



```java
package com.spring.input.logic;

import java.util.ArrayList;

public class UserCatalogImpl implements UserCatalog {

    private UserDao userDao;

    public void setUserDao(UserDao userDao) {
        this.userDao = userDao;
    }

    public void entryUser(User user) {
        this.userDao.create(user);
    }

    public ArrayList<User> getUserList() {
        return this.userDao.findAll();
    }

    public User getUserByUserId(String userId) {
        return this.userDao.findByUserId(userId);
    }
}
```

UserCatalog의 구현 클래스로 UserDao에 관련된 함수body들을 정의

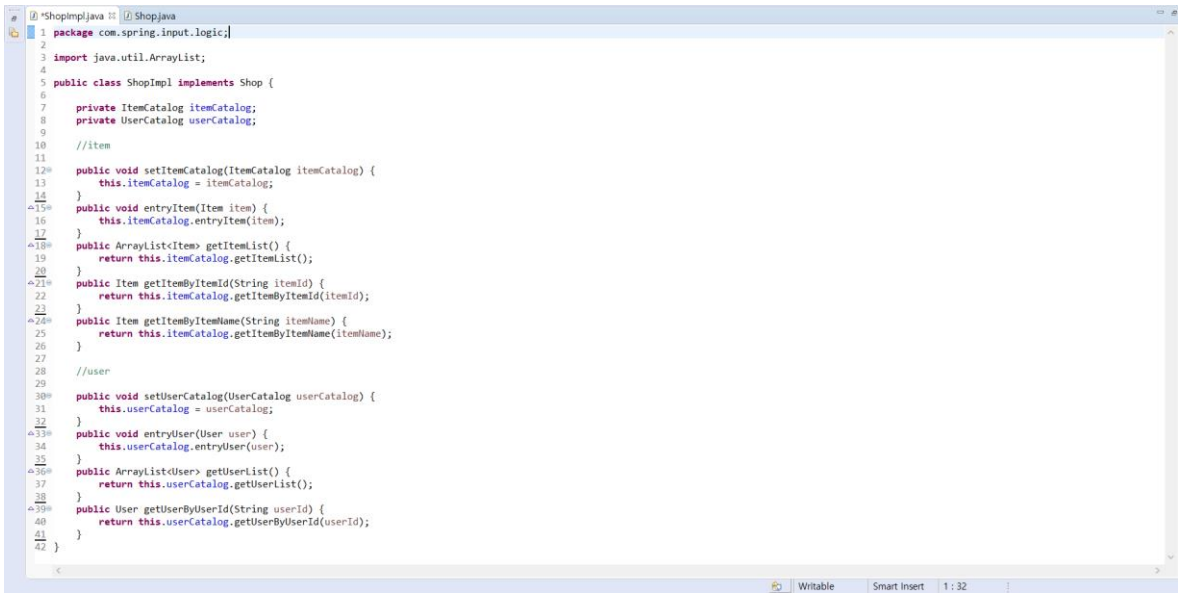
- setUserDao : userDao 연결
- entryUser : 입력받은 유저 userDao에 넘겨줘서 추가
- getUserList : 유저리스트 userDao에서 모두 가져오기
- getUserByUserId : userDao에서 userId로 찾아서 가져오기

Shop

```java
package com.spring.input.logic;

import java.util.ArrayList;

public interface Shop {

    void entryItem (Item item);
    void entryUser (User user);

    ArrayList<Item> getItemList();
    ArrayList<User> getUserList();

    Item getItemByItemId(String itemId);
    Item getItemByItemName(String itemName);
    User getUserByUserId(String userId);
}
```

Interface 클래스로 ItemCatalog, UserCatalog에 관련된 함수이름들을 정의

ShopImpl



```java
package com.spring.input.logic;

import java.util.ArrayList;

public class ShopImpl implements Shop {

    private ItemCatalog itemCatalog;
    private UserCatalog userCatalog;

    //item

    public void setItemCatalog(ItemCatalog itemCatalog) {
        this.itemCatalog = itemCatalog;
    }
    public void entryItem(Item item) {
        this.itemCatalog.entryItem(item);
    }
    public ArrayList<Item> getItemList() {
        return this.itemCatalog.getItemList();
    }
    public Item getItemByItemId(String itemId) {
        return this.itemCatalog.getItemByItemId(itemId);
    }
    public Item getItemByItemName(String itemName) {
        return this.itemCatalog.getItemByItemName(itemName);
    }

    //user

    public void setUserCatalog(UserCatalog userCatalog) {
        this.userCatalog = userCatalog;
    }
    public void entryUser(User user) {
        this.userCatalog.entryUser(user);
    }
    public ArrayList<User> getUserList() {
        return this.userCatalog.getUserList();
    }
    public User getUserByUserId(String userId) {
        return this.userCatalog.getUserByUserId(userId);
    }
}
```

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
- getItemByItemId : itemCatalog에서 itemId로 찾아서 가져오기
- getItemByItemName : itemCatalog에서 itemName으로 찾아서 가져오기
- setUserCatalog : userCatalog 연결
- entryUser : 입력받은 유저 userCatalog에 넘겨줘서 추가
- getUserList : 유저리스트 userCatalog에서 모두 가져오기
- getUserByUserId : userCatalog에서 userId로 찾아서 가져오기

ItemEntryValidator

```java
package com.spring.input.utils;

import com.spring.input.logic.Item;

public class ItemEntryValidator implements Validator {

    public boolean supports(Class<?> clazz) {
        return Item.class.isAssignableFrom(clazz);
    }

    public void validate(Object command, Errors errors) {

        Item user = (Item) command;

        if (!StringUtils.hasLength(user.getItemId())) {
            errors.rejectValue("itemId", "error.required");
        }

        if (!StringUtils.hasLength(user.getItemId())) {
            errors.rejectValue("ItemName", "error.required");
        }

        if (!StringUtils.hasText(user.getDescription())) {
            errors.rejectValue("description", "error.required");
        }

        if (errors.hasErrors()) {
            errors.reject("error.input.user");
        }
    }
}
```

에러 체크 (에러 발생시 messages properties에서 호출하여 출력)

UserEntryValidator

ItemEntryValidator.java *UserEntryValidator.java

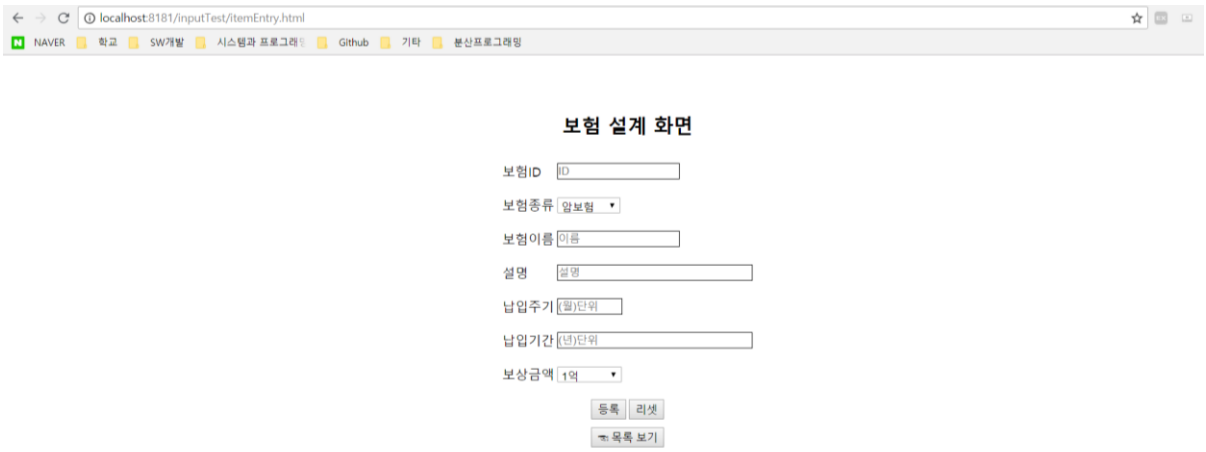
```java
package com.spring.input.utils;

import com.spring.input.logic.User;

public class UserEntryValidator implements Validator {

    public boolean supports(Class<?> clazz) {
        return User.class.isAssignableFrom(clazz);
    }

    public void validate(Object command, Errors errors) {

        User user = (User) command;

        if (!StringUtils.hasLength(user.getUserId())) {
            errors.rejectValue("userId", "error.required");
        }

        if (!StringUtils.hasLength(user.getAge())) {
            errors.rejectValue("age", "error.required");
        }

        if (!StringUtils.hasLength(user.getUserName())) {
            errors.rejectValue("userName", "error.required");
        }

        if (!StringUtils.hasText(user.getSsn())) {
            errors.rejectValue("ssn", "error.required");
        }

        if (!StringUtils.hasText(user.getAddress())) {
            errors.rejectValue("address", "error.required");
        }

        if (!StringUtils.hasText(user.getEmail())) {
            errors.rejectValue("email", "error.required");
        }

        if (errors.hasErrors()) {
            errors.reject("error.input.user");
        }
    }
}
```

에러 체크 (에러 발생시 messages properties에서 호출하여 출력)

3) 실행결과

보험 설계하기

1. 보험정보 입력 후 등록



2. 보험 생성 완료

보험 확정하기

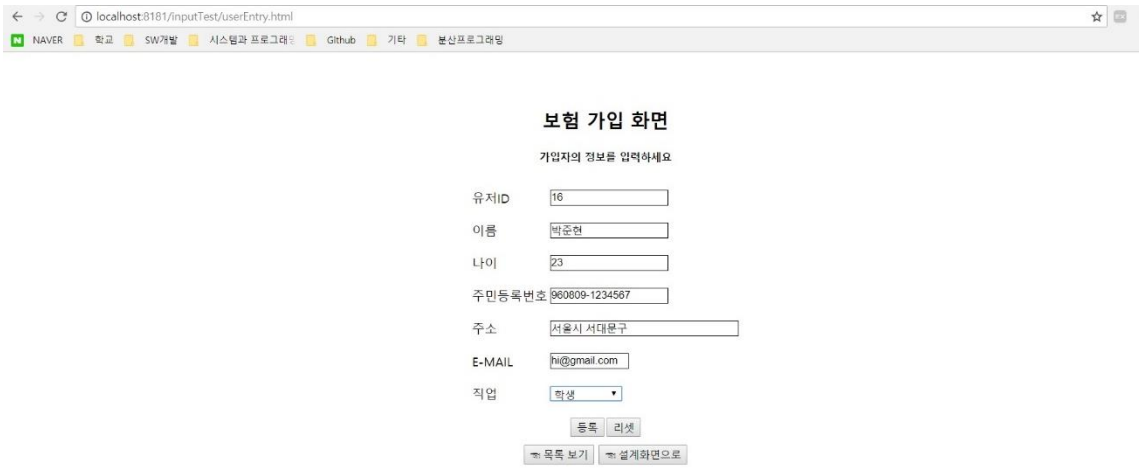
3. 이전에 생성된 여러 보험상품들 중 확정 거절할 보험 선택하여 삭제 (삭제하지 않으면 확정된 상품)

4. 선택하여 확정 거절

보험 가입하기

5. 가입을 원하는 보험의 상세정보 클릭 후, 가입하기 클릭

6. 가입자 정보를 입력 후 등록



7. 가입 완료