SILKROAD 프로젝트

WEB

2019/12/31 00:12 http://blog.naver.com/sevenn0801/221754731233

쥬얼리류를 워낙 좋아하여 크롬하츠 폰트를 응용해서 어플리케이션을 구현해보았음.

I LOV CHROMEHEARTS SO MUCH♥

사용기술 : JAVA, HTML5, JS, JQUERY(이젠 바닐라스크립트덕에 슬슬 놔줘야 될 때가오는것같다..ㅠㅠ),CSS3, AJAX

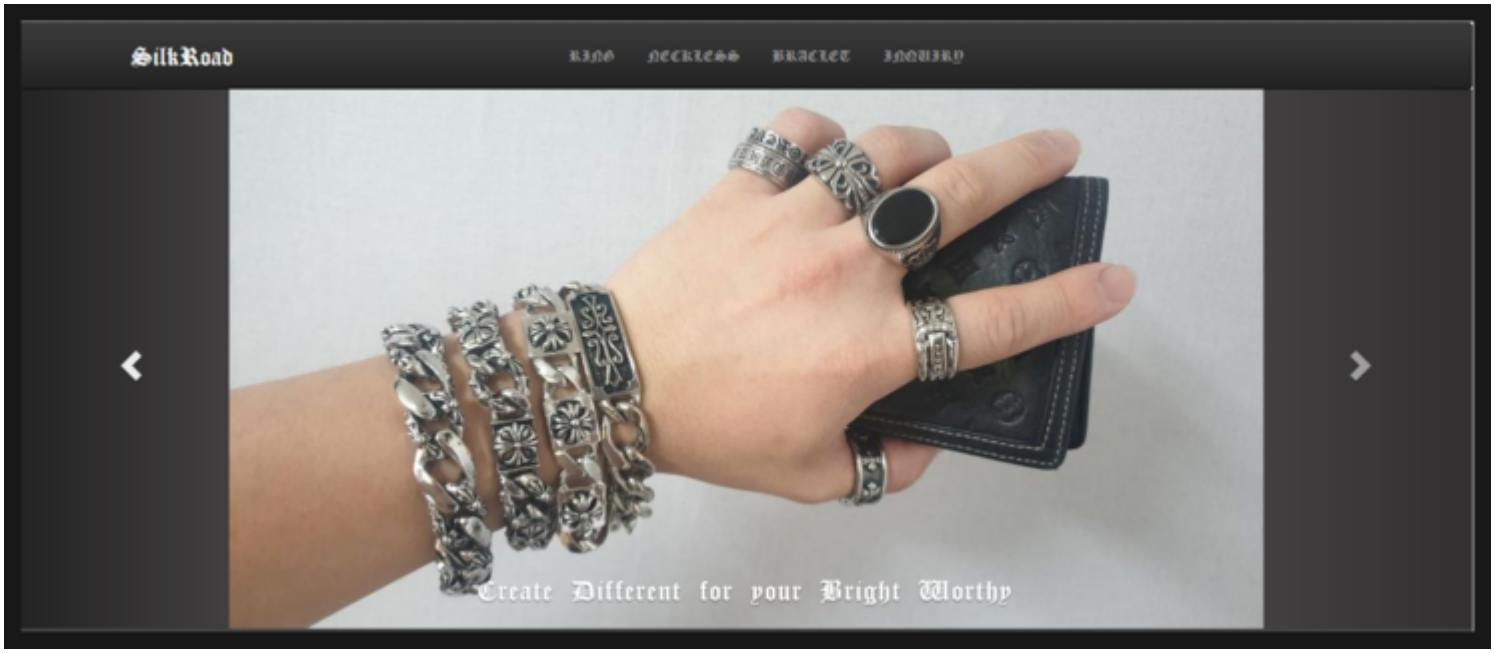
폰트 : 크롬하츠 폰트(Cloisterblack)

PATTERN : Spring MVC

OPTION : Responsive(Media Query, BootStrap)

사용툴 : Eclipse, Visual SVN, MySql, Fiddler

1. *Main.jsp*



데스크탑(좌), 모바일(우)이다. 근데 메뉴가 허전해보이는데...음..

* 원리

스프링 프로젝트는 하나의 어플리케이션으로 취급함으로 서버를 실행할 때 실행할 상대주소를 "/"로 설정하고

web.xml에서 <welcom-file-list>에다가 index.do를 하나 만들어 두었다!

url-mapping 경로방식도 *.do로 받도록 설정하였고,

또한 스프링에서는 컨트롤러에 전달되는 HTTP방식을 GET 방식과 POST방식으로 구분지을수있었는데,

만일 GET방식으로 요청이 올 경우에는 RequestDispatcher 로, POST방식으로 요청이 올 경우에는 Model함수를 사용하여서 main으로 접근하도록 코드를 작성해보았다.

```
<!-- index 설정 -->
<welcome-file-list>
    <welcome-file>index.do</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

```
<!-- URL 경로전달 패턴 설정(디스패처 설정) -->
<servlet>
    <servlet-name>SilkRoad</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/spring/appServlet/SilkRoad-context.xml
        </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>SilkRoad</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

web.xml 일부분

```
// 어플리케이션을 작동하게 될 경우 index를 타고 main.jsp로 로딩 되도록.
// POST
@RequestMapping(value = "/index", method = RequestMethod.POST)
public String home(Model model) {
    logger.info("POST방식으로 접근 시도");
    model.addAttribute("main", "/Template/main.jsp");
    return "index";
}

// GET
@RequestMapping(value = "/index", method = RequestMethod.GET)
public void home(HttpServletRequest request, HttpServletResponse response) throws Exception {
    logger.info("GET방식으로 접근 시도");
    RequestDispatcher fowardMain = request.getRequestDispatcher("/Template/main.jsp");
    fowardMain.forward(request, response);
}
```

컨트롤러

이렇게 코드를 깔아놓고보니 viewResolver가 리턴되는 페이지경로의 suffix를 처리해주는데 GET으로 받을 때 굳이 저렇게 받아야하나 싶다..

이 부분에있어서는 고민을 좀더 해봐야겠다.

그리고 main으로 전달되는 index.jsp!

```
<jsp:forward page="${main}" />
```

index.jsp

이 경로를 타고 main.jsp로 가도록 한다.

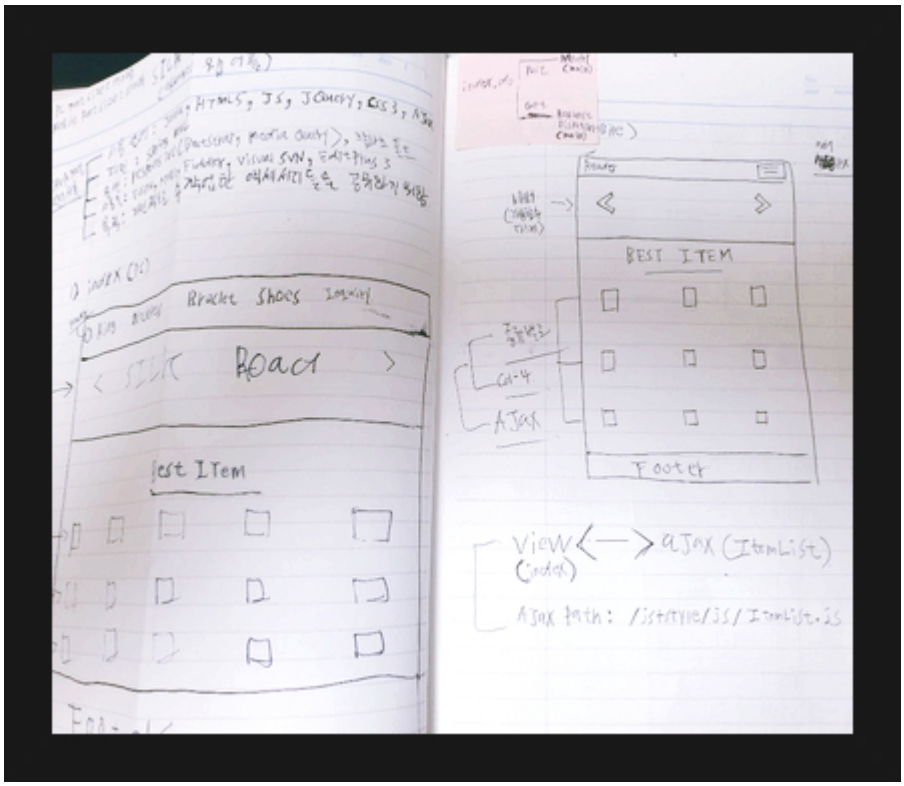
```
<jsp:include page="/Template/Top.jsp" />
    <div class="index-main">
        <div class="itemList"></div>
    </div>
<jsp:include page="/Template/Footer.jsp" />
```

main.jsp

아! 데스크탑과 모바일에서 BestItem 템플릿은 AJAX(비동기통신)를 사용하여 아이템목록을 불러오도록 해놓았다.

이미지는 딱히 어떤 것을 사용할 지몰라서 내가 아주 좋아하는 실버브랜드의 ㅋㄹㅎㅊ제품을 사용을 했다

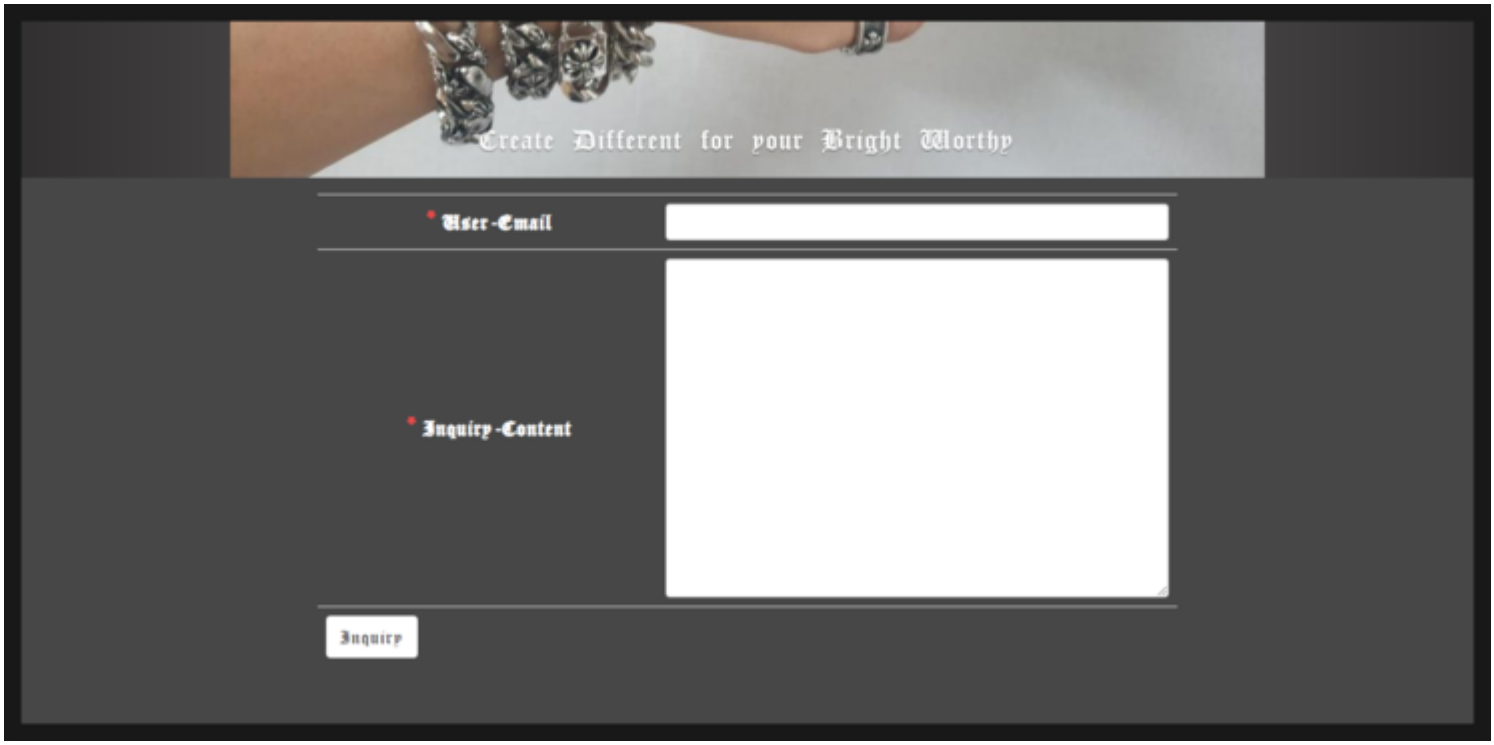
* index(메인) 시안



index 아이디어 스케치 시안

---

2. *Inquiry.jsp*



inquiry 게시판(데스크탑이랑 모바일이 비슷해서 따로 구별하지않았다.)

* 원리

스프링의 기본 데이터 전달방식(view <-> ControllerMapping <-> DAO <-> SERVICE)을 사용해서 만들어보았다.

우선 빨간색 별 표로 표시되어있는 것은 REQUIRED(필수요소)여서 두개의 폼중에 하나라도 입력 되지 않을경우 이거나, 이메일형식에 어긋나는 데이터를 입력할 경우에 진행이 되지않게 설정해두었다.

※View(URL-HANDLER)

```
<li><a href="#" id="inquiryMenu">INQUIRY</a></li>
```

메뉴에서 INQUIRY를 클릭시 기능이 실행이된다.

```
// 메뉴에서 문의게시판으로 이동할 경우
$("#inquiryMenu").on("click", function() {
    window.location.href = "/Board/Inquiry.do";
});
```

↕

※Controller(Mapping)

```
/*
 * 문의 게시판(RequestToViewNameTranslator)
 */
@RequestMapping("/Board/Inquiry.do")
public Map<String, Object> inquiryBoard() {
    HashMap<String, Object> BoardMap = new HashMap<String, Object>();
    BoardMap.put("", "");
    return BoardMap;
}
```

컨트롤러에서 문의게시판(VIEW) 경로 매핑부분

```
  /*
   * 문의저장기능(검토대상 1)
   */
  @RequestMapping("/sendInquiry.do")
  public String BoardFunction(@ModelAttribute("vo") SilkRoadVO vo) {
      boolean inquiry = service.insertInquiry(vo);
      if (inquiry) {
          logger.info("문의글 문의 성공~");
      }
      return "/Board/function/BoardFunction";
  }
```

관리자DB에 문의 데이터를 남기도록 하는 기능

↕

※Service

```
@Autowired
private SilkRoadDAOImpl dao;

public static SilkRoadServiceImpl getServiceImpl() {
    return new SilkRoadServiceImpl();
}


@Override
public boolean insertInquiry(SilkRoadVO vo) {
    // TODO Auto-generated method stub
    return dao.insertInquiry(vo);
}
```

컨트롤러에서 MODEL로서 적용될 서비스단(Dependency-Injection)

↕

※DAO

```
public String sqlMap(String sqlMapper) {
    String sql = "";
    switch (sqlMapper) {
    case "insert":
        sql = "insert into board(BoardWriter, BoardContent) values(?,?)";
        break;
    }
    return sql;
}
```

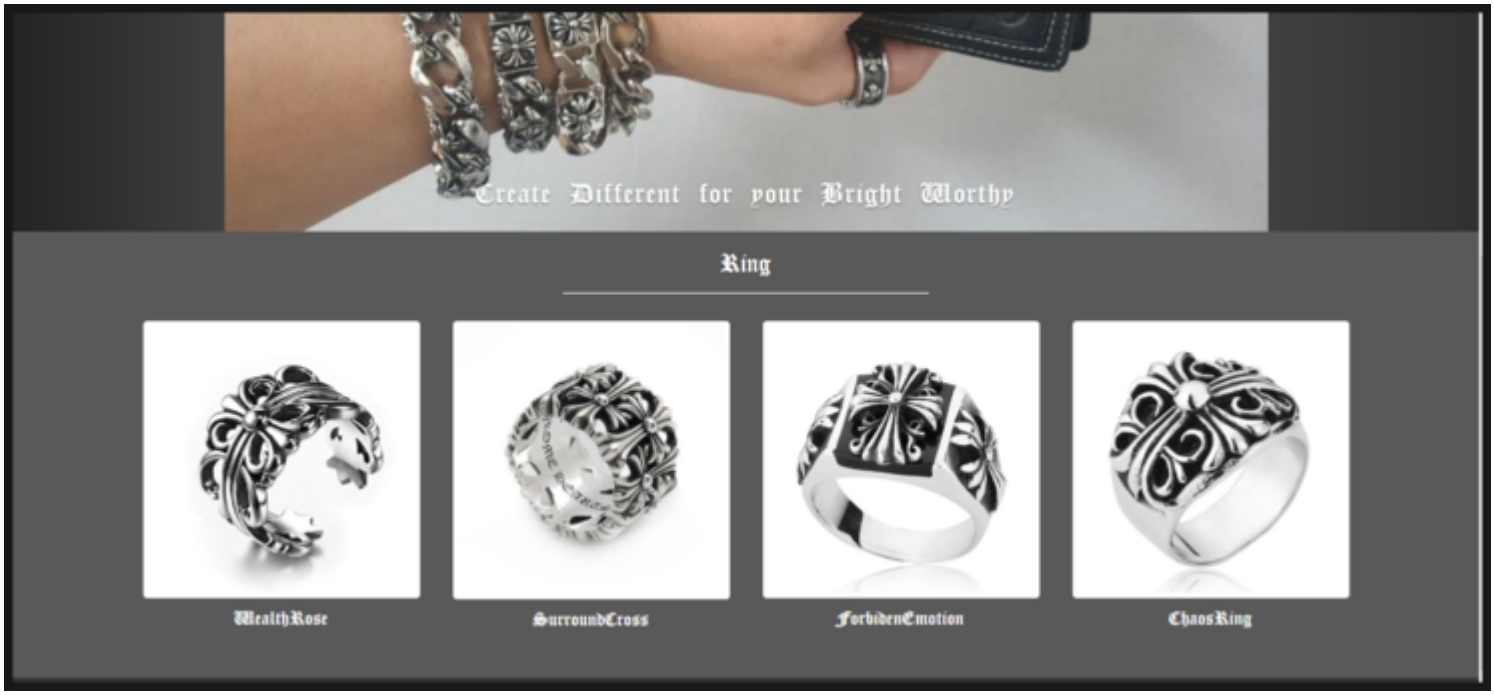
sqlMapper 파라미터의 종류(insert, delete, update...)에 따른 sql문 생성 함수

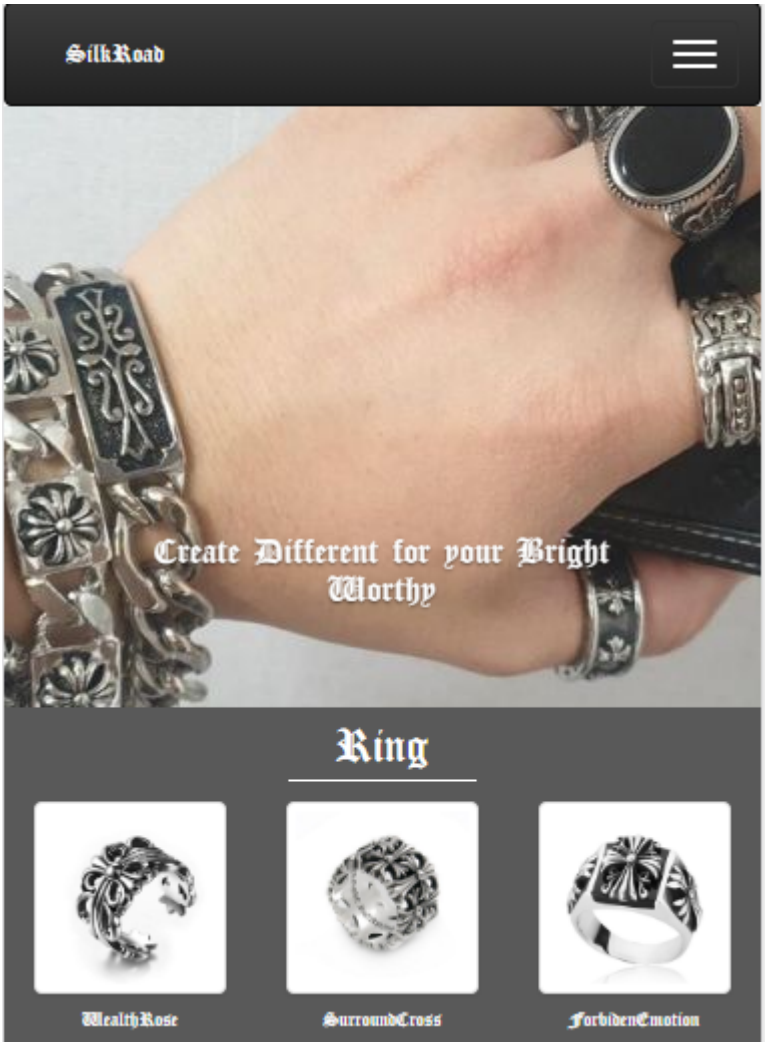
```java
@Override
public boolean insertInquiry(SilkRoadVO vo) {
    boolean check = false;
    String sql = "insert";

    try {
        conector = DBConnection.getDBConnector();
        sqlState = conector.prepareStatement(super.sqlMap(sql));

        sqlState.setString(1, vo.getClientEmail());
        sqlState.setString(2, vo.getClientInquiry());
        if (sqlState.executeUpdate() > 0) {
            check = true;
        }
    } catch (Exception e) {
        logger.error(e.toString());
    } finally {
        try {
            if (conector != null) {
                conector.close();
            }
            if (sqlState != null) {
                sqlState.close();
            }
        } catch (Exception e2) {
            // TODO: handle exception
        }
    }
    return check;
}
```

문의게시글 데이터를 관리자가 받을 수있도록하는 DAO(Dependency-Injection)

---

3. ring, Bracelet, Necklace.jsp(Categories)





데스크탑(좌), 모바일(우) 반지류 카테고리(다른 품목도 동일.)

*원리

메뉴에 있는 메뉴목록을 클릭하면 해당하는 페이지 경로로 이동한다.

이 기능도 컨트롤러를 통해서 어느 페이지로 이동을 해야하는지 결정이된다.

※View(URL-HANDLER)

```
<div class="col-md-8 col-xs-9 collapse navbar-collapse" id="Menu">
    <ul class="nav navbar-nav navbar-right">
        <li><a href="/categories.do?pageInfo=ring">RING</a></li>
        <li><a href="/categories.do?pageInfo=neckless">NECKLESS</a></li>
        <li><a href="/categories.do?pageInfo=braclet">BRACLET</a></li>
        <li><a href="#" id="inquiryMenu">INQUIRY</a></li>
    </ul>
</div>
```

↕

※Controller(Mapping)

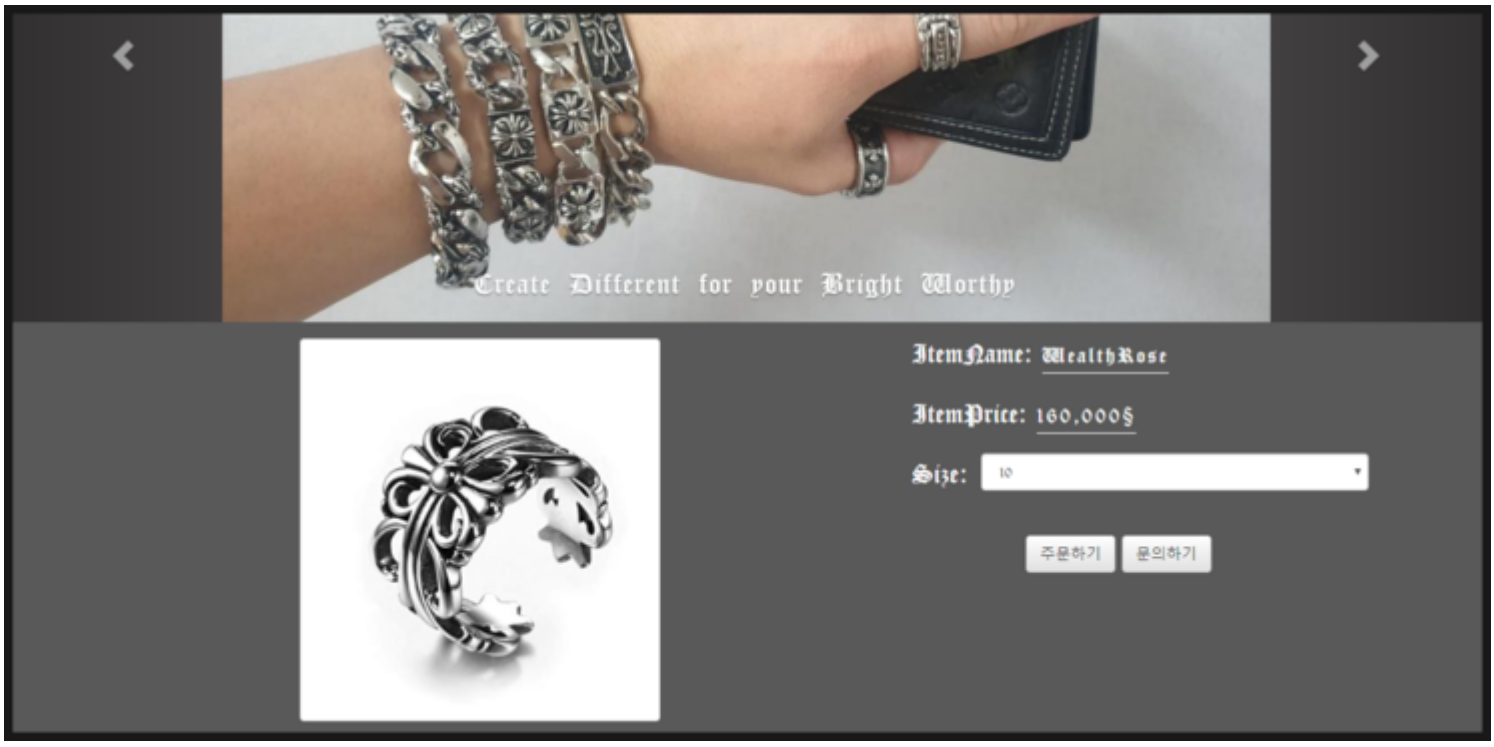
```
/*
 * 카테고리별 페이지 이동
 */
@RequestMapping("/categories.do")
public String Categories(@RequestParam(value = "pageInfo") String categories) {
    String CategoryPage = "";
    switch (categories) {
    case "ring":
        CategoryPage = "/Categories/ring";
        break;
    case "neckless":
        CategoryPage = "/Categories/necklace";
        break;
    case "braclet":
        CategoryPage = "/Categories/braclet";
        break;
    }
    return CategoryPage;
}
```

또한, 아이템 품목 박스들은 현재 프로젝트폴더에 등록되어있는 사진의 수 만큼 FOR문을 돌려 출력한다.

이곳에서 아이템 품목은 본 해당 아이템의 코드를 생성한다.

```
<c:forEach begin="1" end="4" var="item">
    <div class="col-md-3 col-xs-4">
        <img src="/picture/Bracelet/Bracelet${item}.jpg" id="ringImg" class="img-responsive img-thumbnail" />
        <c:if test="${item == 1}">
            <a href="/Categories/Detail/category.do?item=bracelet1" class="item-introduce">
                <span class="item-span">Cross</span>
            </a>
        </c:if>
        <c:choose>
            <c:when test="${item == 2}">
                <a href="/Categories/Detail/category.do?item=bracelet2" class="item-introduce">
                    <span class="item-span">SurroundCross</span>
                </a>
            </c:when>
            <c:when test="${item == 3}">
                <a href="/Categories/Detail/category.do?item=bracelet3" class="item-introduce">
                    <span class="item-span">ForbidenEmotion</span>
                </a>
            </c:when>
            <c:when test="${item == 4}">
                <a href="/Categories/Detail/category.do?item=bracelet4" class="item-introduce">
                    <span class="item-span">ChaosRing</span>
                </a>
            </c:when>
        </c:choose>
    </div>
</c:forEach>
```

그 후에, 아무 아이템을 클릭 해서 URL을 타고 들어가면~



이렇게 해당 아이템 사진과 아이템이름, 가격, 주문자의 착용 사이즈가 출력 된다.(자, 어떻게 작동되는지 코드를 한번 뜯어봅시다.)

```java
// (RequestToViewNameTranslator)
@RequestMapping("/Categories/Detail/category.do")
public Map<String, Object> categoryItem(@RequestParam(value = "item") String ItemKind) {
    HashMap<String, Object> CategoryMap = new HashMap<String, Object>();

    String test = super.ItemDetail(ItemKind);
    logger.info("선택아이템명: " + test);

    // 아이템별 종류 식별
    CategoryMap.put("ItemKind", test);
    // 아이템 코드
    CategoryMap.put("ItemCode", ItemKind);

    return CategoryMap;
}
```

아이템을 클릭할경우, 고유코드를 super.ItemDetail(아이템코드)로 전송하는 함수

```java
// ItemCode변수로 가져온 값에 따른 상품 이름 배분 함수
public String ItemDetail(String ItemCode) {
    String name = "";
    // 반지류
    if (ItemCode.indexOf("ring") > -1) {
        // 아이템코드명
        String[] ItemCodes = { "ring1", "ring2", "ring3", "ring4" };
        for (String Items : ItemCodes) {
            // 일치하는 아이템코드값이 존재할 경우
            if (Items.equals(ItemCode)) {
                name = CodeResolver(Items);
            }
        }
    }

    // 팔찌류
    else if (ItemCode.indexOf("bracelet") > -1) {
        String[] ItemCodes = { "Bracelet1", "Bracelet2", "Bracelet3", "Bracelet4" };
        for (String Items : ItemCodes) {
            if (Items.equals(ItemCode)) {
                name = CodeResolver(Items);
            }
        }
    }
    // 넥레이스류
    else if (ItemCode.indexOf("necklace") > -1) {
        String[] ItemCodes = { "necklace1", "necklace2", "necklace3", "necklace4" };
        for (String Items : ItemCodes) {
            if (Items.equals(ItemCode)) {
                name = CodeResolver(Items);
            }
        }
```

ItemCode로 전달받은 값이 ItemCodes안에 존재하는 고유값과 일치할 경우,

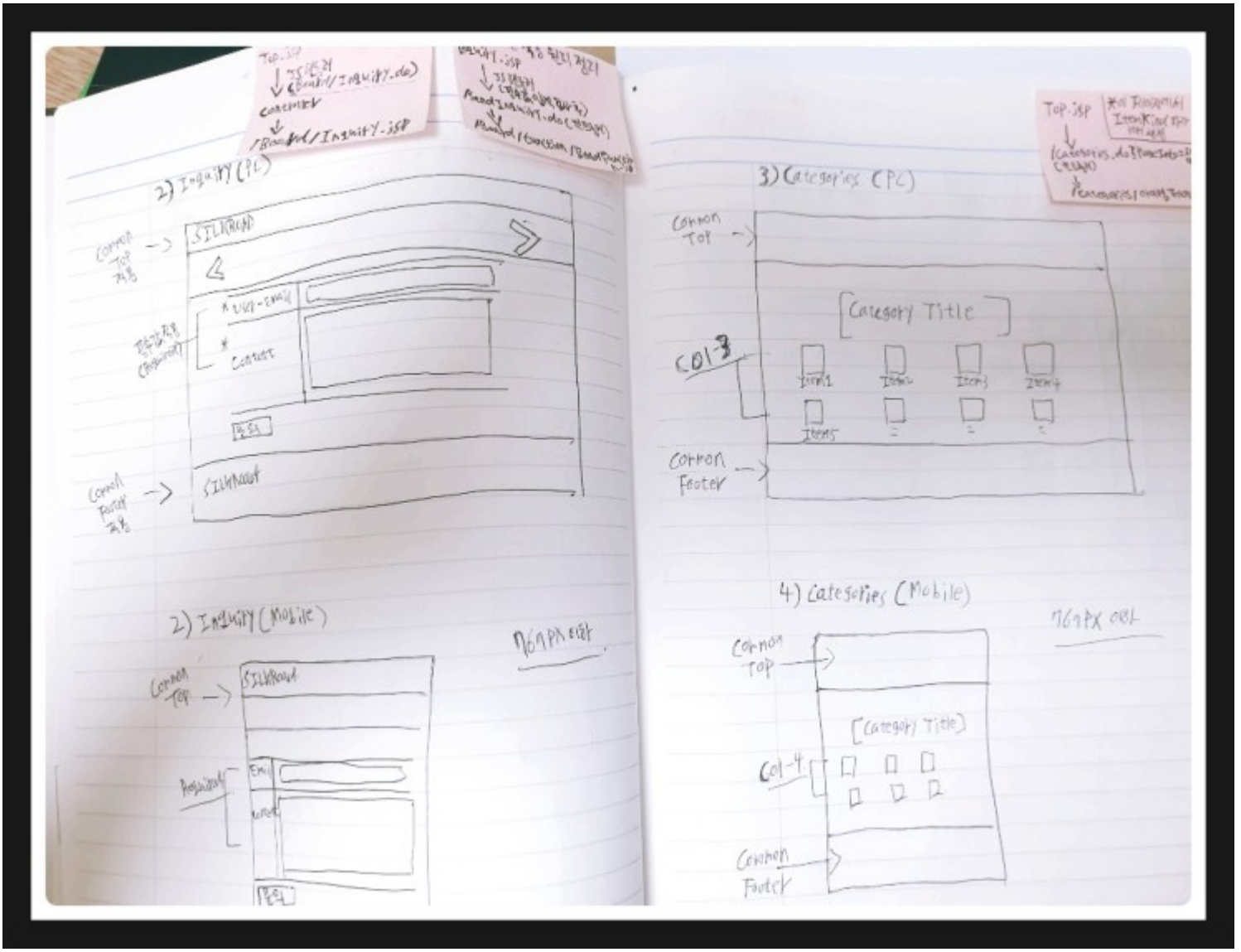
```java
// 아이템코드해석용 함수(1차)
public String CodeResolver(String ItemCode) {
    String ItemName = "";
    if (ItemCode.equals("ring1")) {
        ItemName = "WealthRose";
    } else if (ItemCode.equals("ring2")) {
        ItemName = "SurroundCross";
    } else if (ItemCode.equals("ring3")) {
        ItemName = "ForbidenEmotion";
    } else if (ItemCode.equals("ring4")) {
        ItemName = "ChaosRing";
    }else {
        ItemName = "제작중";
    }
    return ItemName;
}
```

CodeResolver(아이템코드)함수에 집어넣어서 아이템명을 받아오도록한다.

```jsp
<div class="ItemInfo-wrapper row">
    <div class="col-md-7 col-xs-12">
        <div class="img-wrapper">
            <c:if test="${fn:indexOf(ItemInfo, 'ring') != -1}">
                <img src="/picture/ring/${ItemInfo}.jpg" class="img-responsive img-thumbnail" />
            </c:if>
            <c:choose>
               <c:when test="${fn:indexOf(ItemInfo, 'Bracelet') != -1}">
                   <img src="/picture/Bracelet/${ItemInfo}.jpg" class="img-responsive img-thumbnail" />
               </c:when>
               <c:when test="${fn:indexOf(ItemInfo, 'necklace') != -1}">
                  <img src="/picture/necklace/${ItemInfo}.jpg" class="img-responsive img-thumbnail" />
               </c:when>
            </c:choose>
        </div>
    </div>

    <div class="clearfix visible-xs"></div>

    <div class="col-md-5 col-xs-12">
        <div class="describe-wrapper">
            <form method="post" id="ItemDetailFrm">
                <ul class="describe-ul form-group">
                    <%--컨트롤러에서 SilkRoadItemDetail.java를 돌려서 가져온 아이템명 --%>
                    <li>ItemName: <span>${ItemKind}</span></li>
                    <li>ItemPrice: <span>160,000§</span></li>
                    <li>Size: <select class="form-control size-form"
                        name="sizeForm" id="size-form">
                            <c:forEach var="size" begin="10" end="28">
                                <option value="${size}" class="size-option">${size}</option>
                            </c:forEach>
                    </select>
```

그러고나서, 컨트롤러에서 아이템명과 아이템코드 데이터를 아이템정보출력페이지(category.jsp)로 전달한다.

* Inquiry(문의게시판) & ItemCategories(아이템 종류별) 시안



Inquiry & ItemCategories 아이디어 스케치 시안