인터넷 서점 최종 보고서

작성자 - 서주원

1. 프로젝트 기획

요구사항 분석

- **사용자 요구사항 분석**

1) 물건을 구매할 때 물건의 실물을 보고 싶다.
2) 내가 구매했던 물건들의 목록을 보고싶다.
3) 게시판에 관리자에게 필요한 것을 요구하고 싶다.
4) 게시판을 통해서 다른 사람들과 의사소통을 하고싶다.

프로젝트 설계

- 프로그램 이름 : 다담아
- 프로그램 목적 : 사이트에 올려진 책을 구매하고 구매품에 대한 정보를 다른 회원들과 공유하며 상호 피드백을 할 있는 플랫폼 제공

- 개발 언어 : JAVA, HTML5, CSS3, JSP, Firebase, Servlet

2. 제공 기능 설명

제공 기능 설계

- 제공 기능 설명

1) 관리자가 상품을 등록할 수 있는 기능 제공

2) 관리자가 상품을 관리 할 수 있는 기능 제공

3) 관리자가 고객을 삭제할 수 있는 기능 제공

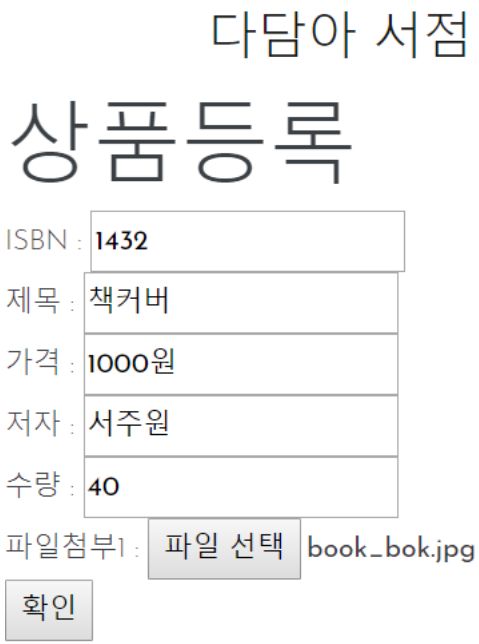
4) 후기 게시판을 통해서 물건에 대한 정보를 타 고객과 상호 교류하는 기능 제공

5) 고객이 물품을 구매하고 구매한 물품의 목록을 볼 수 있는 기능 제공

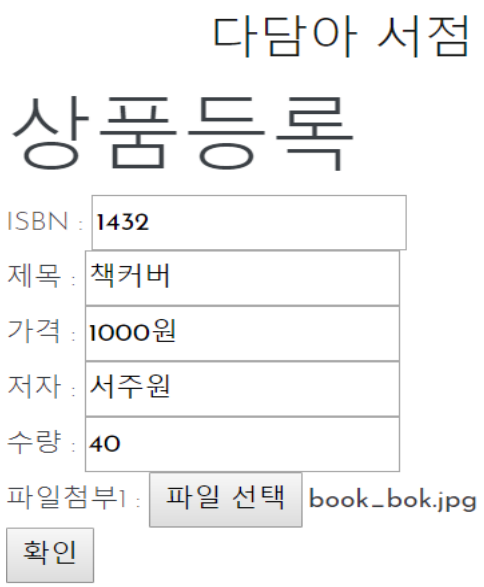
첫 번째 기능 설계

- **관리자가 상품을 등록하는 기능**

1. **상품등록 버튼을 누른다.**
2. **상품에 대한 내용(사진, 내용, 가격)을 입력한다.**
3. **등록완료 버튼을 누른다.**
4. **등록된 상품이 상품리스트에 나타난다.**



첫 번째 기능 구현(1)




```
<!-- SECTION -->
  <section>

    <div>
        <h1>상품등록</h1>
    </div>
    <form method="post" enctype="multipart/form-data">
    ISBN : <input type="text" id="isbn"><br>
    제목 : <input type="text" id="title"><br>
    가격 : <input type="text" id="price"><br>
    저자 : <input type="text" id="author"><br>
    수량 : <input type="text" id="amount"><br>
    파일첨부1 : <input type="file" id="file1"><br>

    <input type="button" value="확인" onclick="bookLoad()">
    </form>

  </section>
```

상품의 대한 정보를 form에 입력한 후 버튼을 통해 bookLoad()함수 실행

이 때, 이미지 업로드를 위해 form에 enctype="multipart/form-data" 선언

첫 번째 기능 구현(2)

```
function bookLoad(){

    var isbn = document.getElementById("isbn").value;
    var title = document.getElementById("title").value;
    var price= document.getElementById("price").value;
    var author= document.getElementById("author").value;
    var amount= document.getElementById("amount").value;
    var path = document.getElementById("file1").value;

    var filename= path.substring(12,path.length);
```

Form에서 입력한 value 값들을 각 변수에 저장한다.

이때, 이미지의 경로에서 image의 file과 확장자만을 알아내기 위해서
Substring이라는 함수를 이용했다.

첫 번째 기능 구현(3)

```
    var book_data = firebase.database().ref('book_data/'+isbn);
    book_data.set({
          isbn: isbn
     });

    var book_info = firebase.database().ref('book_info/'+isbn);
    book_info.set({
      title: title,
      price: price,
      author: author,
      amount: amount,
      src: filename,
      isbn : isbn
    });

     alert("책 등록이 완료되었습니다.");
     location.reload();
}
```

입력한 value 중 isbn이라는 value를 주 키로 book_data라는 table에 data를 넣어준다.

또 isbn value를 주 키로 book_info라는 Table에 책 상품에 대한 제목, 작가, 가격, 사진 등의 정보를 삽입 시켜준다.

책 등록이 완료되었다는 문구를 띄우고 해당 itempLoad.jsp를 reload(새로고침) 시켜서 입력 창을 item input form을 깨끗하게 만들어 준다.

두 번째 기능 설계

- 관리자가 상품을 관리 하는 기능

1. 상품관리 버튼을 누른다.
2. 현재 등록된 상품들이 나열된다.
3. 상품을 보고 삭제할 상품을 고른다.
4. 삭제 버튼을 누른다.
5. 해당 책이 삭제된 후의 상품 리스트가 출력된다.

두 번째 기능 구현(1)

ISBN	제목	저자	가격	남은수량	사진경로	관리
1	개념원리2	이홍석	100000원	0		삭제
2	RPM	이홍섭	20000원	120		삭제
7	7	7	7	7		삭제

```
<!-- SECTION -->
  <section>
     <table id="infotable">
        <tr>
           <td>ISBN</td>
           <td>제목</td>
           <td>저자</td>
           <td>가격</td>
           <td>남은수량</td>
           <td>사진경로</td>
           <td>관리</td>
        </tr>
     </table>
  </section>
```

현재 등록된 상품들을 확인하는 동적 table을 만들기 위한 HTML table구조

두 번째 기능 구현(2)

```
allBookdata();

function allBookdata()
{
   var book_data = firebase.database().ref('book_data');
   book_data.once('value', function(snapshot) {
      snapshot.forEach(function(childSnapshot) {
         var tmp = childSnapshot.val();
         allBookInfo(tmp.isbn);
      });
   });
}
```

itemManage.jsp page에 들어오면
allBookdata() 함수를 실행시킨다.

allBookdata() 함수에서 book_data에 저장된
각 각의 snapshot에 대해서 그 snapshot의 속성인
Isbn을 인자로 넣어 allBookInfo(tmp.isbn)함수를
실행시킨다.

두 번째 기능 구현(3)

```
function allBookInfo(isbn)
{
   table = document.getElementById("infotable");
   var book_info = firebase.database().ref('book_info/' + isbn);
   book_info.once('value', function(snapshot) {
      var tmp = snapshot.val();
```

Isbn을 parameter값으로 받아서

Book_info table에서 isbn의 값이 parameter의 isbn과 같은 snapshot에 대해서 해당 snapshot의 정보

즉, 책의 정보를 추출한다.

두 번째 기능 구현(4)

```
table = document.getElementById("infotable");
new_tr = document.createElement("tr");

td_isbn = document.createElement("td");
td_isbn.innerHTML = isbn;
new_tr.appendChild(td_isbn);

td_title = document.createElement("td");
td_title.innerHTML = tmp.title;
new_tr.appendChild(td_title);


td_src = document.createElement("td");
td_src.innerHTML= "<img src='./img/"+tmp.src+"' width=50 height=50>";
new_tr.appendChild(td_src);


del_btn = document.createElement("input");
del_btn.setAttribute("type", "button");
del_btn.setAttribute("onclick", "del_click(" + isbn + ")");
del_btn.setAttribute("value", "삭제");
new_tr.appendChild(del_btn);

table.appendChild(new_tr);
```

Table을 infotable이라는 id를 가진 html form에 대해서 동적으로 생성해 주는 코드다.

앞에 book_info의 snapshot에서 추출한 정보들을 동적으로 만들어지는 table의 td값에 알맞게 넣어준다.

이때, 각 상품에 대해서 이미지를 출력해 주기 위해서 다음 코드와 같은 형태를 취해준다.

setAtrribute를 이용해 del_click(isn)함수를 실행하는 삭제 버튼을 동적으로 생성한다.

두 번째 기능 구현(5)

```
function del_click(isbn)
{
   var book_data = firebase.database().ref('book_data/' + isbn);
   book_data.remove();

   var book_info = firebase.database().ref('book_info/' + isbn);
   book_info.remove();


   alert("해당 책을 삭제하였습니다.");
   location.reload();

}
```

Isbn을 parameter로 가지는 함수다.
Book_data, book_info table에서 isbn의 값을
키로 가지는 entitty를 .remove()를 사용해서 제거한다.

이후에 location.reload();를 이용해 itemManage.jsp를
재시작 시켜서 눈에 보이는 상품 리스트를
Update 시켜준다.

세 번째 기능 설계

- 관리자가 고객을 삭제하는 기능

1. 고객관리 버튼을 누른다.
2. 현재 등록된 고객들이 나열된다.
3. 리스트를 보고 삭제할 고객을 고른다.
4. 삭제 버튼을 누른다.
5. 해당 고객이 삭제된 후의 고객 리스트가 출력된다.

세 번째 기능 구현(1)

ID	이름	학년	전공	성별	관심사	접속시간	관리
11111	서주원	4학년	컴퓨터공학	남자	야구	2019-06-02 21:32:57	삭제
22222	김상현	3학년	컴퓨터공학	남자	군대	2019-06-02 21:41:11	삭제
44444	4	4학년	4	여자	4	2019-06-02 23:45:17	삭제
zzzzz	관리자	1학년	관리자	남자	관리자	2019-06-03 00:30:40	삭제

```
<!-- SECTION -->
  <section>
     <table id="infotable">
        <tr>
           <td>ISBN</td>
           <td>제목</td>
           <td>저자</td>
           <td>가격</td>
           <td>남은수량</td>
           <td>사진경로</td>
           <td>관리</td>
        </tr>
     </table>
  </section>
```

현재 등록된 고객들을 확인하는 동적 table을 만들기 위한 HTML table구조

세 번째 기능 구현(2)

```
allUserdata();


function allUserdata()
{
   var user_data = firebase.database().ref('user_data');
   user_data.once('value', function(snapshot) {
      snapshot.forEach(function(childSnapshot) {
         var tmp = childSnapshot.val();
         allUserProfile(tmp.user_id, tmp.last_login);
      });
   });
}
```

admin.jsp page에 들어오면
allUserdata() 함수를 실행시킨다.

allUserdata() 함수에서 user_data에 저장된
각 각의 snapshot에 대해서 그 snapshot의 속성인
User_id와 last_login을 인자로 넣어
allUserProfile(tmp.user_id, tmp.last_login)함수를
실행시킨다.

세 번째 기능 구현(3)

```
function allUserProfile(_id, time)
{
   table = document.getElementById("infotable");
   var user_data = firebase.database().ref('user_profile/' + _id);
   user_data.once('value', function(snapshot) {
      var tmp = snapshot.val();
```

_id, time을 paramete값으로 받아서
User_profile table에서 id의 값이 parameter의 id와 같은 snapsho에 대해서 해당 snapsho의 정보
즉, 회원가입 시 작성한 고객의 profile를 추출한다.

세 번째 기능 구현(4)

```
table = document.getElementById("infotable");
new_tr = document.createElement("tr");

td_id = document.createElement("td");
td_id.innerHTML = _id;
new_tr.appendChild(td_id);

td_name = document.createElement("td");
td_name.innerHTML = tmp.user_name;
new_tr.appendChild(td_name);



del_btn = document.createElement("input");
del_btn.setAttribute("type", "button");
del_btn.setAttribute("onclick", "del_click(" +_id + ")");
del_btn.setAttribute("value", "삭제");
new_tr.appendChild(del_btn);

table.appendChild(new_tr);
```

Table을 infotable이라는 id를 가진 html form에 대해서 동적으로 생성해 주는 코드다.

앞에 user_profile의 snapshot에서 추출한 정보들을 동적으로 만들어지는 table의 td값에 알맞게 넣어준다.

setAtrribute를 이용해 del_click(id)함수를 실행하는 삭제 버튼을 동적으로 생성한다.

세 번째 기능 구현(5)

```
function del_click(_id)
{

   var user_data = firebase.database().ref('user_data/' + _id);
   user_data.remove();

   var user_profile = firebase.database().ref('user_profile/' + _id);
   user_profile.remove();

   var user_post = firebase.database().ref('user_post/' + _id);
   user_post.remove();


   var user_get = firebase.database().ref('user_get/'+_id);
   user_get.remove();


   alert("대상 사용자를 삭제하였습니다.");
   location.reload();

}
```

Id를 parameter로 가지는 함수다.

User_data, profile, post, get 등의 table의 id와
Parameter의 _id와 같은 user에 대해서
.remove()를 사용해 해당 user에 대한 정보를 제거한다.

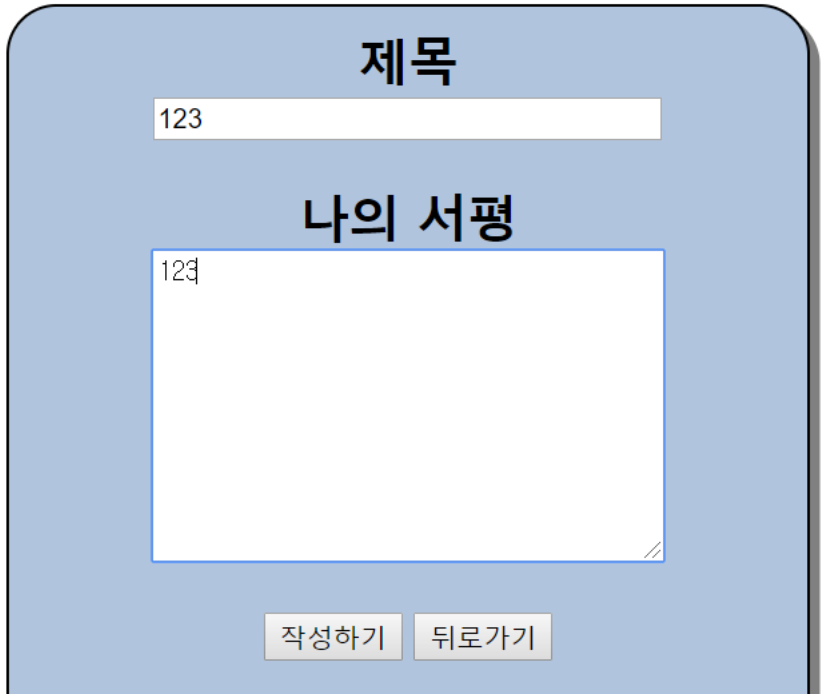
이후에 location.reload();를 이용해 admin.jsp를
재시작 시켜서 눈에 보이는 고객 리스트를
Update 시켜준다.

네 번째 기능 설계

- 후기 게시판을 통해서 물건에 대한 정보를 타 고객과 상호 교류하는 기능 제공

1. 게시판에 들어간다.
2. 글을 쓰는 페이지로 넘어가 글을 작성한다.
3. 게시판에 글을 작성 시 작성자와 작성 날짜가 나오도록 게시판에 표시한다.

네 번째 기능 구현(1)




```
<div id="post">

    <label class="jua" style="font-size:25px;">제목</label><br>
    <input type="text" id="post_title" class="jua"><br><br>
    <label class="jua" style="font-size:25px;">나의 서평</label><br>
    <textarea id="post_content" class="jua"></textarea><br><br>
    <input type="button" value="작성하기" onclick="writeP()" class="jua btn btn-secondary">
    <input type="button" value="뒤로가기" onclick="goBack()" class="jua btn btn-secondary">

</div>
```

포스트를 작성하기 위한 HTML table구조

작성하기 버튼을 눌러 writeP()함수 실행

네 번째 기능 구현(2)

```
/* 포스트 작성 */
function writeP()
{

    var post_title = document.getElementById("post_title").value;
    var post_content = document.getElementById("post_content").value;
    var post_time = getTimeStamp();
    var post_date = getDateStamp();
    var user_post = firebase.database().ref('user_post/' + '<%= id%>');

    // create user_post
    user_post.set
    ({
        user_id : id,
        post_title : post_title,
        post_content : post_content,
        post_date : post_date,
        post_time : post_time
    });

    alert("포스트가 작성되었습니다.");
    history.back();
}
```

Form에 작성한 포스트의 내용들을
변수에 담아온다.

User_post에 현재 로그인 되어있는 id를 키로
작성한 post의 title, content, date, time을
User_post table에 삽입한다.

포스트가 작성되었다는 문구를 띄운다.
게시판 즉 board.jsp 화면으로 되돌아 간다.

네 번째 기능 구현(3)

```
var user_post = firebase.database().ref('user_post');
    user_post.on('value', function(snapshot) {
    snapshot.forEach(function(childSnapshot) {
    var tmp = childSnapshot.val();
```

User_post에 있는 모든 id에 대해서 접근한다.

```
// DB로 부터 내용 가져옴
h3.innerHTML = tmp.post_title;
p.innerHTML = "작성자 : " + tmp.user_id;
p.innerHTML += "<br>" + "나의서평 : " + tmp.post_content + "<br>"
                  + "작성시간 : " + tmp.post_date + " "+ tmp.post_time;
```

각 id를 키로 한 childSnapshot들의 정보 Title, user_id, content, date, time의 정보를 가져와 알맞은 element에 삽입시켜준다..

네 번째 기능 구현(4)

```
// 동적으로 element 생성
var div = document.createElement("div");
var h3 = document.createElement("h3");
var p = document.createElement("p");
var hr = document.createElement("hr");
```

게시판에 작성될 포스트들의 element를 동적으로 생성한다.

```
div.appendChild(h3);
div.appendChild(p);
div.appendChild(hr);
document.getElementById("container").appendChild(div);
});
```

만들어진 동적 element들을 id가 container인 Form에 삽입 시켜준다.

다섯 번째 기능 설계

- 고객이 물품을 구매하고 구매한 물품의 목록을 볼 수 있는 기능 제공

1. 물품을 구매한다.
2. Profile로 들어간다.
3. 자신의 정보 및 자신이 구매한 물건을 확인한다.

다섯 번째 기능 구현(1)

제목	사진	저자	가격	남은수량	구매
개념원리2		이홍석	100000원	0	구매하기
RPM		이홍섭	20000원	120	구매하기
7		7	7	7	구매하기

```
<section>

   <table id="infotable">
      <tr>


         <td>제목</td>
         <td>사진</td>
         <td>저자</td>
         <td>가격</td>
         <td>남은수량</td>
         <td>구매</td>
      </tr>
   </table>
</section>
```

상품 구매를 위한 HTML table구조

다섯 번째 기능 구현(2)

```
if(tmp.amount<=0){
    buy_btn = document.createElement("input");
    buy_btn.setAttribute("type", "button");
    buy_btn.setAttribute("onclick", "nobuy()");
    buy_btn.setAttribute("value", "구매하기");
    new_tr.appendChild(buy_btn);

}

else{
    buy_btn = document.createElement("input");
    buy_btn.setAttribute("type", "button");
    buy_btn.setAttribute("onclick", "buy_click("+isbn + ")");
    buy_btn.setAttribute("value", "구매하기");
    new_tr.appendChild(buy_btn);
}
```

만약 상품의 개수가 0개 이하이면 Nobuy()함수를 실행

상품의 개수가 1이상으로 유효하다면 buy_click(isbn) 함수를 실행

다섯 번째 기능 구현(3)

```
function buy_click(isbn)
{
    var ret = confirm("구매하시겠습니까?");
    if(ret)
    {
        var edit_book_info = firebase.database().ref('book_info/' + isbn);
        edit_book_info.once('value', function(snapshot) {
        var tmp = snapshot.val();

        var _title = tmp.title;
        var _author = tmp.author;
        var _price = tmp.price;
        var _src = tmp.src;
        var _amount = tmp.amount;
```

구매를 하는 것이 맞는 지 confirm을 이용해서 확인한다.

구매를 하는 것이라면 book_info table의 parameter의
Isbn과 같은 isbn snapshot을 찾아서
Title, author, price 등의 책의 정보를

각각의 변수에 삽입해 준다.

다섯 번째 기능 구현(4)

```
        edit_book_info.update({

            title : _title,
            author : _author,
            price : _price,
            src : _src,
            amount : _amount-1
        });

        getBuy(isbn, _title, _author, _price, _src);
        alert("해당 책을 구매하였습니다.");
        });
        location.reload();
    }
    else{
        alert("구매가 취소되었습니다");
    }

}

function nobuy(){
    alert("구매할 수 없습니다!");
```

앞에서 찾은 책에 대해서
Title, author 등의 정보는 같게 해주지만
Amount의 양은 1개 감소시켜서 book_info의
값들을 update 시켜준다.

그 후 getBuy() 함수를 호출한다.
getBuy()가 끝난 후 itemList.jsp를 reload시킨다..

Comfirm에서 취소 시 구매 취소 문구를 띄운다.

Nobuy() 함수 : 구매 할 수 없다는 문구를 띄운다.

다섯 번째 기능 구현(5)

```
function getBuy(isbn, title, author, price, src){

    var user_get = firebase.database().ref('user_get/' + id);
    user_get.set
    ({
        get_isbn : isbn,
        buy_time : getTimeStamp()
    });


}
```

구매한 책에 대한 정보에 대해서 parameter로 받는다.

User_get이라는 table에 현재 접속해 있는 id를 주 키로

구매한 책의 isbn번호
구매시점의 시간

의 값들을 table에 삽입 시켜준다.

다섯 번째 기능 구현(6)

구매한 도서

제목	사진	저자	가격	구매시간
RPM		이홍섭	20000원	2019-06-03 03:02:38

```
getUser();
function getUser(){

    var user_get =firebase.database().ref('user_get/'+id);

    user_get.on('value', function(snapshot) {
        var tmp = snapshot.val();

            findBook(tmp.get_isbn, tmp.buy_time);

      });
}
```

Profile 진입 시 getUser() 함수 호출

User_get table에서 현재 로그인 된 id를 주 키로 가지는 snapshot의 정보를 추출

findBook()함수의 인자로 snapsho의 isbn과 buy_time Value를 paramete로 넘김

다섯 번째 기능 구현(7)

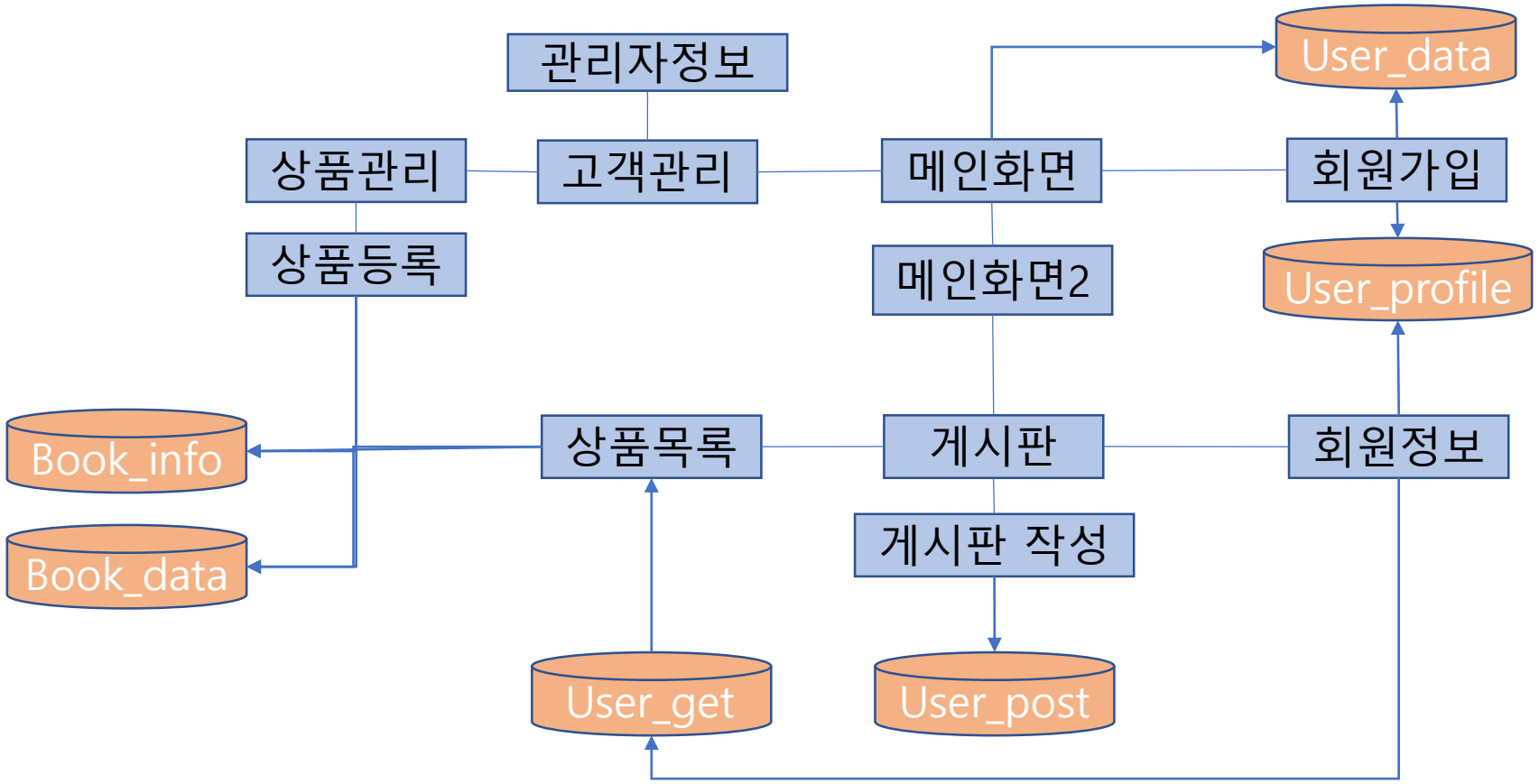
```
function findBook(is_isbn,is_time){
    |
     table = document.getElementById("booktable");
    var book_info = firebase.database().ref('book_info/'+is_isbn);

    book_info.on('value', function(snapshot) {
        var tmp = snapshot.val();


        new_tr = document.createElement("tr");

        td_title = document.createElement("td");
        td_title.innerHTML = tmp.title;
        new_tr.appendChild(td_title);

        td_src = document.createElement("td");
        td_src.innerHTML= "<img src='./img/"+tmp.src+"' width=50 height=50>";
        new_tr.appendChild(td_src);
```

User_get의 isbn값을 가지는 책에 대해서 Book_info table에서 같은 isbn값을 가지는 책의 정보를 추출한다.

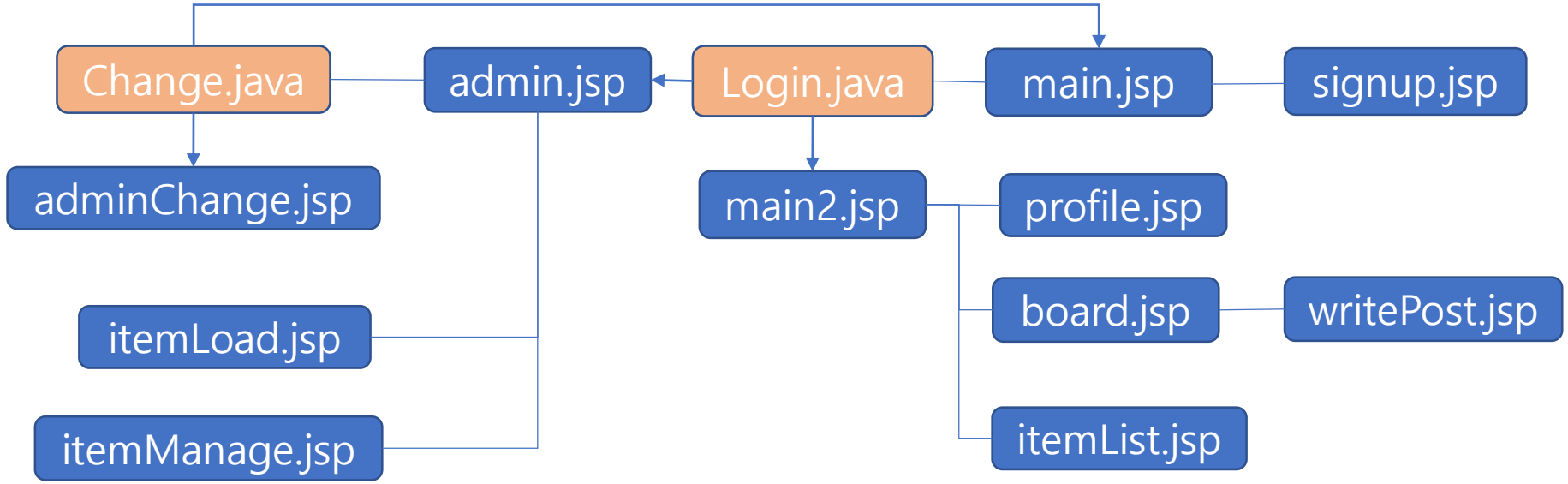
그 이후 동적으로 element를 만들어서 화면에 출력해 준다.

3. 웹 페이지 구성

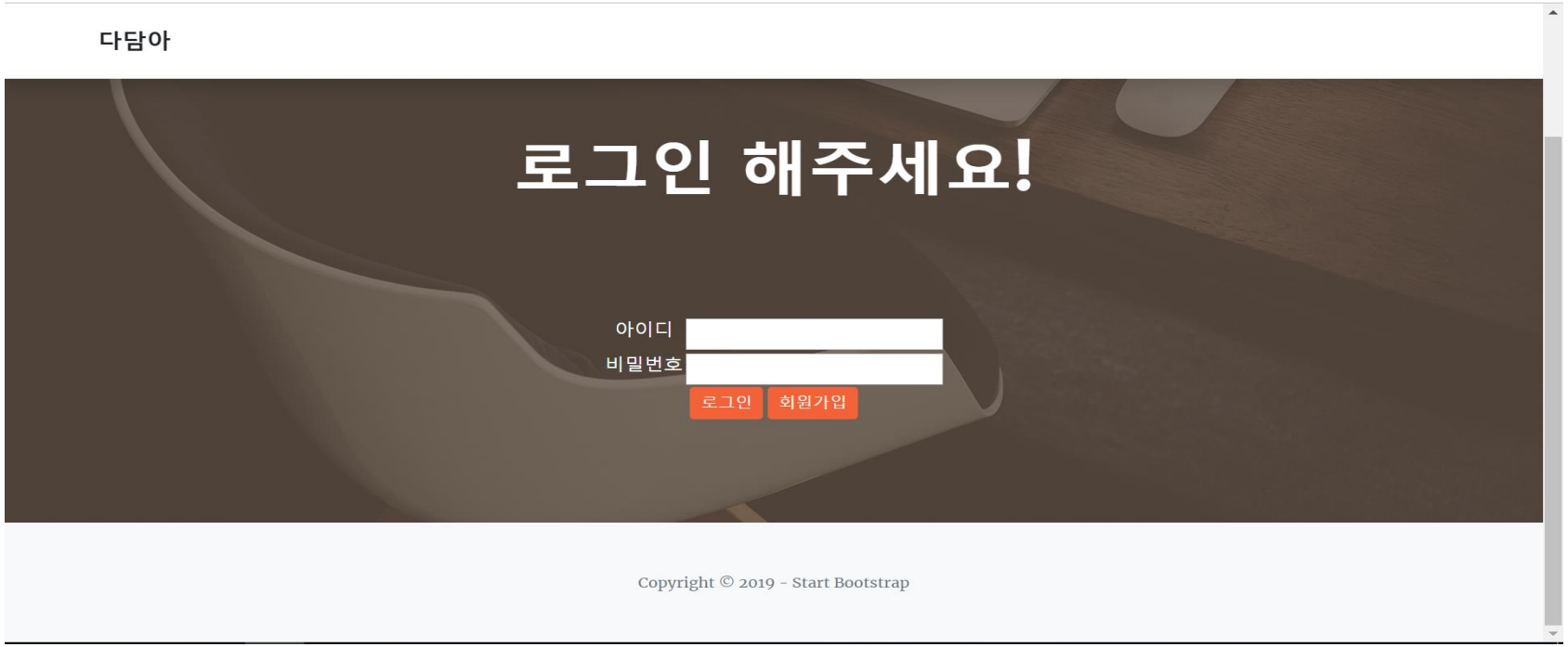
아키텍쳐


관리자정보
User_data
상품관리
고객관리
메인화면
회원가입
상품등록
메인화면2
User_profile
Book_info
상품목록
게시판
회원정보
Book_data
게시판 작성
User_get
User_post

웹 페이지 구성


Change.java
admin.jsp
Login.java
main.jsp
signup.jsp
adminChange.jsp
main2.jsp
profile.jsp
itemLoad.jsp
board.jsp
writePost.jsp
itemManage.jsp
itemList.jsp

Main.jsp(로그인 전)



로그인을 하는 부분입니다.
로그인 버튼을 누르면 Login.java에서 관리자인지 아닌지를 판단해서
관리자 페이지 또는 고객 페이지로 넘겨줍니다.

Signup.jsp

회원가입을 하는 부분입니다.
성별과 학년 부분은 text로 입력하기 보다는 버튼이 더 간편해 radio type으로 만들었습니다.
중복확인 버튼을 눌러 아이디 field가 unable상태가 되어야 완료 버튼을 누를 수 있습니다.

main2.jsp

다담아 서점

새로 출시된 책들

개념원리

고객으로 로그인하면 넘어오는 페이지 입니다.
새로 출시된 책들을 fotorama를 이용해 자동으로 넘어가면서
여러 책들의 이미지를 보여주게 만들었습니다.

itemList.jsp

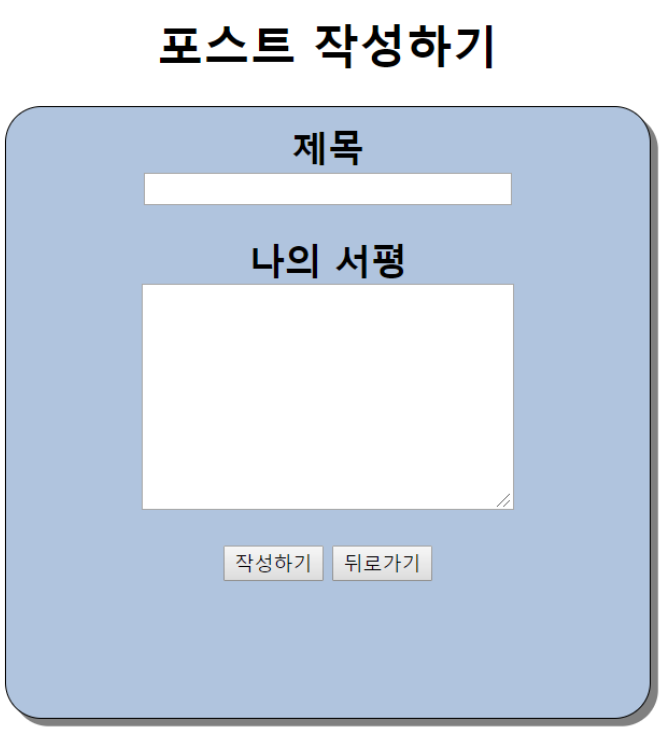
다담아 서점

HOME 상품 목록 게시판 마이페이지 llllll님 반갑습니다! 로그아웃

제목	사진	저자	가격	남은수량	구매
개념원리2		이홍석	100000원	0	구매하기
RPM		이홍섭	20000원	119	구매하기
7		7	7	7	구매하기

책을 구매할 수 있는 페이지입니다.
남은 수량이 0이면 구매할 수 없습니다.
구매하기 버튼을 누르면 구매를 진행할 수 있습니다.

writePost.jsp

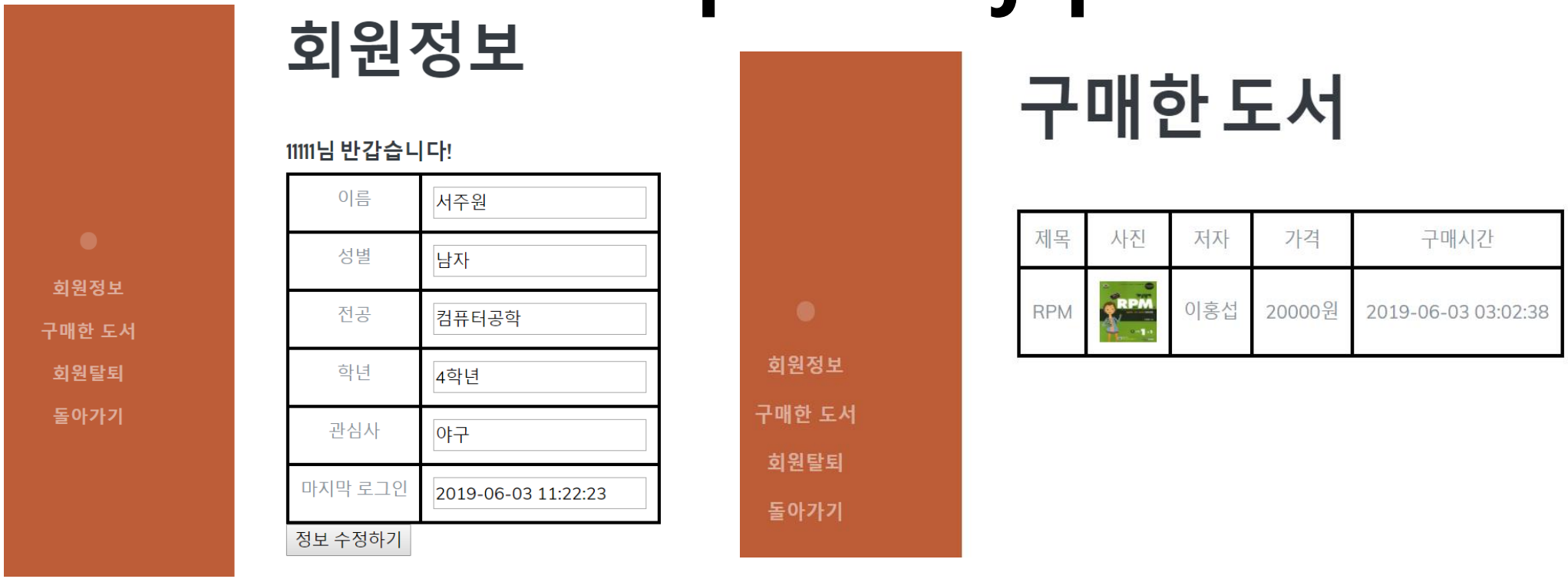


포스트를 작성할 수 있는 부분입니다.
작성하기 버튼을 누르면 작성이 되었다는 문구가 나옵니다.
그 후 게시판 페이지로 넘어갑니다.

board.jsp

게시판 페이지 입니다.
앞에 post작성 페이지에서 사람들이 작성한 글에 대해서 볼 수 있는 페이지입니다.

profile.jsp



내 프로필을 보는 부분입니다.
가입할 때 작성했던 회원 정보를 확인 및 수정할 수 있습니다.
내가 구매한 도서를 확인할 수 있습니다.
회원탈퇴를 할 수 있습니다.
왼쪽에 탭 버튼으로 원하는 기능을 클릭 시 해당 section을 보여줍니다.

admin.jsp

다담아 서점

비밀번호를 입력하세요. 고객관리 상품등록 상품관리 관리자님 반갑습니다! 로그아웃

정보확인

ID	이름	학년	전공	성별	관심사	접속시간	관리
11111	서주원	4학년	컴퓨터공학	남자	야구	2019-06-03 11:22:23	삭제
22222	김상현	3학년	컴퓨터공학	남자	군대	2019-06-03 01:47:34	삭제
44444	4	4학년	4	여자	4	2019-06-02 23:45:17	삭제
zzzzz	관리자	1학년	관리자	남자	관리자	2019-06-03 11:29:04	삭제

관리자로 로그인하면 바로 보이는 페이지입니다.
고객들을 정보를 확인하고 삭제할 수 있는 기능이 있습니다.
고객 table위에 비밀번호를 입력해 정보확인 버튼을 누르면
Chagne.java에 정보가 넘어가 비밀번호가 맞으면 관리자 정보확인 페이지로,
하지만 강제 로그아웃이 되어 main.jsp 로 넘어갑니다.

itemLoad.jsp

상품을 등록하는 부분입니다.
등록하는 상품에 정보를 입력하고 상품의 이미지를 업로드 시킬 수 있습니다.
확인버튼을 누르면 입력한 상품이 등록됩니다.

ItemManage.jsp

다담아 서점

고객관리 상품등록 상품관리 관리자님 반갑습니다! 로그아웃

ISBN	제목	저자	가격	남은수량	사진경로	관리
1	개념원리2	이홍석	100000원	0		삭제
2	RPM	이홍섭	20000원	119		삭제
7	7	7	7	7		삭제

상품을 관리하는 부분입니다.
상품의 내용이 이상하거나 판매자 측에서 내려 달라고 요청할 시
바로 삭제할 수 있게 모든 상품에 대해서 삭제할 수 있게 만들었습니다.

adminChange.jsp

다담아 서점

고객관리 상품등록 상품관리 관리자님 반갑습니다! 로그아웃

이름	서주원
성별	남자
전공	컴퓨터공학
학년	3학년
관심사	야구
마지막 로그인	그런거 없음

관리자의 정보를 확인하는 부분입니다.

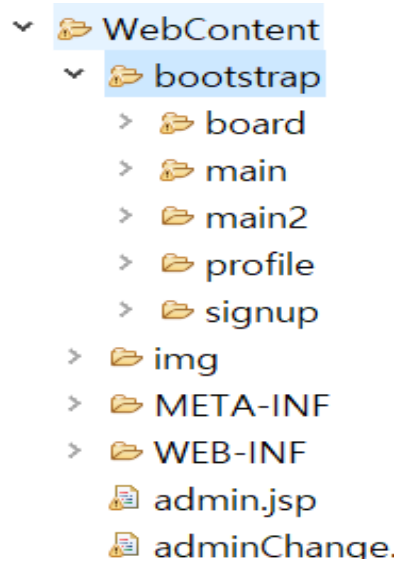
4. 화면 디자인

Bootstrap 사용(1)

```
<!-- bootstrap code -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integ
   <!-- Font Awesome Icons -->
  <link href="bootstrap/main/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">

  <!-- Google Fonts -->
  <link href="https://fonts.googleapis.com/css?family=Merriweather+Sans:400,700" rel="stylesheet">
  <link href='https://fonts.googleapis.com/css?family=Merriweather:400,300,300italic,400italic,700,700italic

  <!-- Plugin CSS -->
  <link href="bootstrap/main/vendor/magnific-popup/magnific-popup.css" rel="stylesheet">

  <!-- Theme CSS - Includes Bootstrap -->
  <link href="bootstrap/main/css/creative.min.css" rel="stylesheet">
```

디자인을 적용하고 싶은 페이지 마다 해당 코드를 import시켜서
화면을 디자인 했습니다.

Bootstrap 사용(2)




```
<!-- Plugin CSS -->
<link href="bootstrap/main/vendor/magnific-popup/magnific-popup.css" rel="stylesheet">
```

디자인 파일 관리를 편리하게 하기 위해서 bootstrap폴더를 따로 만들고
디자인 필요한 부분별로 하위 폴더를 만들었습니다.
그 후, bootstrap을 이용할 때 오른쪽 사진의 bootstrap/main/ 처럼
영역별로 bootstrap에 접근할 수 있게 만들었습니다.

Bootstrap 원본(1)

ABOUT

EXPERIENCE

EDUCATION

SKILLS

INTERESTS

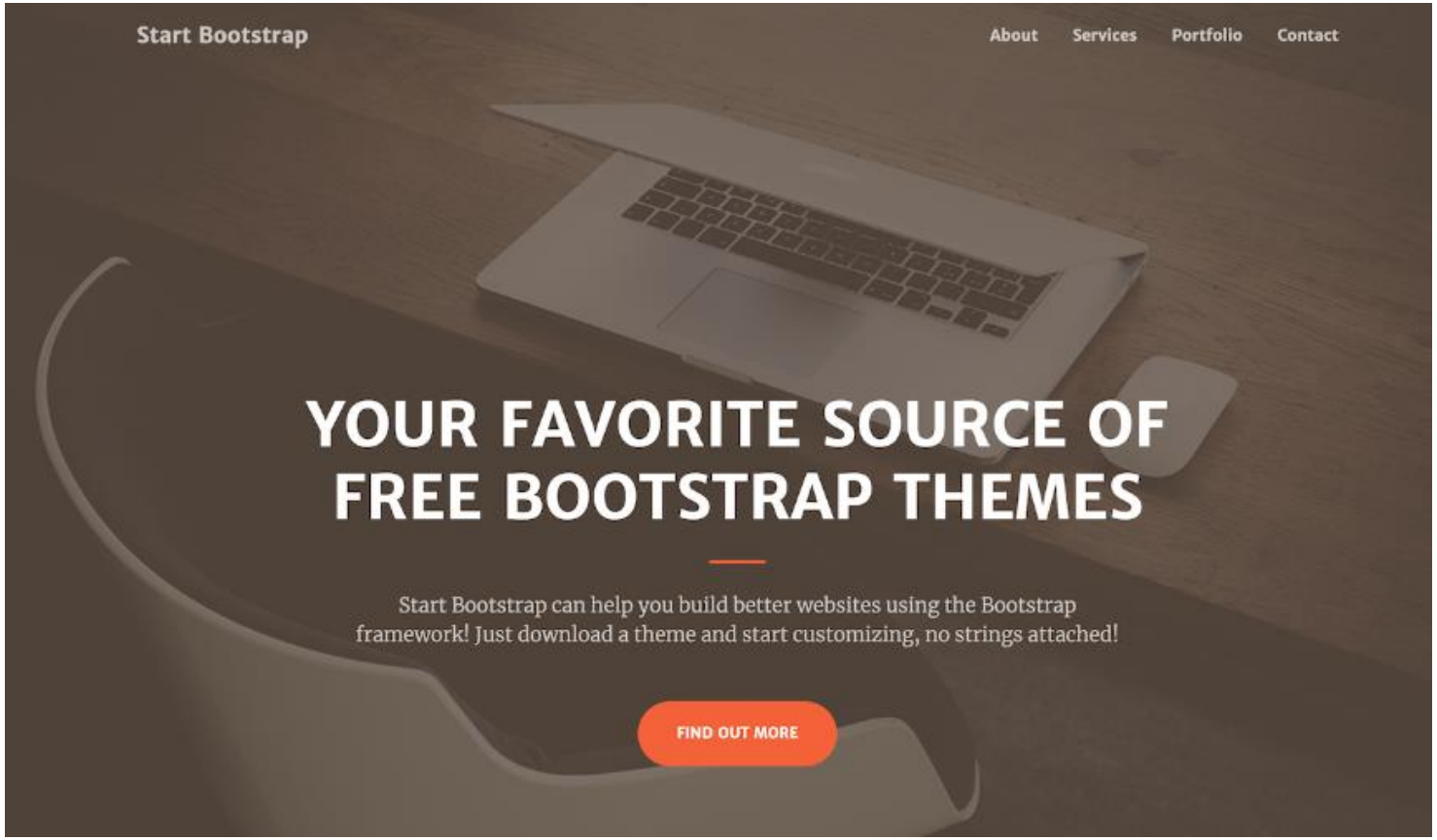
AWARDS

CLARENCE TAYLOR

3542 BERRY STREET · CHEYENNE WELLS, CO 80810 · (317) 585-8468 · NAME@EMAIL.COM

I am experienced in leveraging agile frameworks to provide a robust synopsis for high level overviews. Iterative approaches to corporate strategy foster collaborative thinking to further the overall value proposition.

Bootstrap 원본(2)



Bootstrap 원본(3)

Fotoroma 사용(1)

```
<!-- 포토로마 -->
    <link  href="https://cdnjs.cloudflare.com/ajax/libs/fotorama/4.6.4/fotorama.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/fotorama/4.6.4/fotorama.js"></script>
<article>

    <div class="fotorama"
            data-loop = "true"
            data-autoplay="2000"
            data-arrows="false"
            data-keyboard="true"
            data-width="100%"
            data-height="400"
            data-click="false"
            data-swipe="false"
            data-nav="false"
            >
        <div data-img="bootstrap/main2/images/book_genum.jpg">
            <a href="#" onclick="goItemlist()">
                    개념원리
            </a>
        </div>
        <div data-img="bootstrap/main2/images/book_rpm.jpg">
                <a href="#" onclick="goItemlist()">
                RPM
            </a>
        </div>
        <div data-img="bootstrap/main2/images/book_su.jpg">
                <a href="#" onclick="goItemlist()">
                    수능특강
                </a>
        </div>
    </div>
</article>
```

포토로마 CDN을 위의 사진처럼 가져와서 이용했습니다.

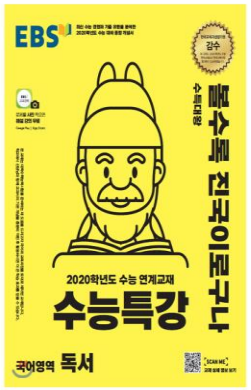
Fotoroma 사용(2)

다담아 서점

HOME 상품 목록 게시판 마이페이지 zzzzz님 반갑습니다! 로그아웃

새로 출시된 책들

수능특강

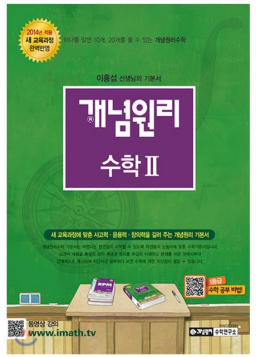


다담아 서점

HOME 상품 목록 게시판 마이페이지 llllll님 반갑습니다! 로그아웃

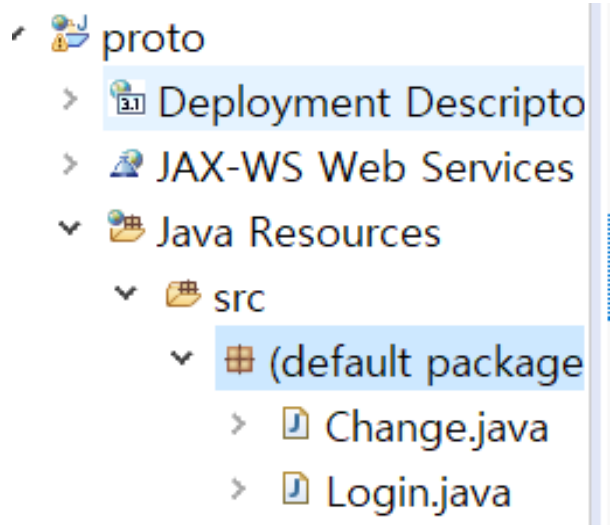
새로 출시된 책들

개념원리



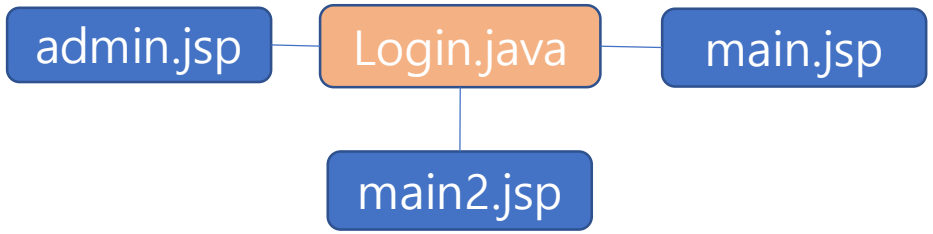
5. 서블렛 사용

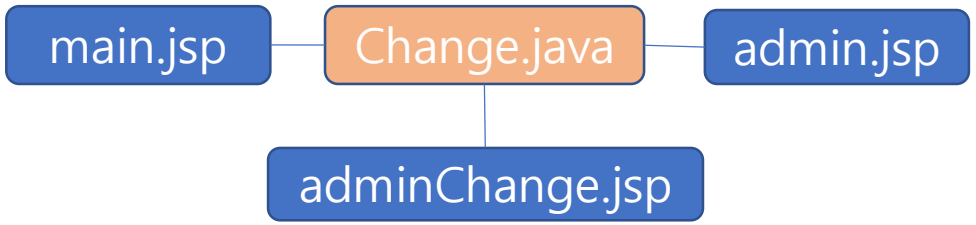
서블렛



서블렛은 총 2개로
Login.java, Change.java가 있습니다.

서블렛의 역할





Main.jsp에서 들어온 정보를 확인해서
관리자이면 admin.jsp로
고객이면 main2.jsp로
페이지를 넘겨줍니다.

Admin.jsp에서 들어온 정보를 확인해서
비밀번호가 맞다면 adminChange.jsp로
틀리다면 main.jsp로
페이지를 넘겨줍니다.

Login.java(1)

```
<!-- Login.java로 보내는 code -->
        <form id="form1" method="POST">
            <table>
                <tr>
                    <td>
                        <label for="id" style="color:white">아이디
                    </td>
                    <td>
                        <input type="text" name="id" id="id" size="20px">
                    </td>
                </tr>
                <tr>
                    <td>
                        <label for="pw" style="color:white">비밀번호
                    </td>
                    <td>
                        <input type="password" name="pw" id="pw" size="20px">
                    </td>
                </tr>
            </table>
        <input type="button" class="btn btn-primary btn-sm" onclick="myjsplogin()" value="로그인">
        <input type="reset" class="btn btn-primary btn-sm" onclick="gosignup()"  value="회원가입">
        </form>
```

```
var form = document.getElementById("form1");
form.setAttribute("action", "./Login?id="+id+"&&pw="+pw+"");
form.submit();// form 제출 -> Login.java로 이동 -> Servlet에 의해 처리
```

Main.jsp의 form에 대해서 받은 값들을 submit 해주어 Login.java로 값을 넘겨줍니다.

Login.java(2)

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    // TODO Auto-generated method stub

    HttpSession session = request.getSession();
    String id = request.getParameter("id");
    String pw = request.getParameter("pw");


    session.setAttribute("id", id);
    session.setAttribute("pw", pw);

    // 관리자 계정으로 로그인
    if(id.equals("zzzzz") && pw.equals("12345"))
    {
        String admin = "admin.jsp";
        request.getRequestDispatcher(admin).forward(request, response);
    }
    // 일반 계정으로 로그인
    else
    {
        String login = "main2.jsp?user_id=" + id;
        request.getRequestDispatcher(login).forward(request, response);
    }
}
```

Main.jsp에서 form에 쓰인 method방식대로 값을 받아 doGet이 호출됩니다.
이 서블렛은 login하는 값이 유저인지 관리자인지 파악해서 알맞은 사이트로 옮겨줍니다.
getRequestDispatcher()를 통해서 처리 속도를 향상시켰습니다.

Login.java(3)

```
<welcome-file-list>
  <welcome-file>main.jsp</welcome-file>
</welcome-file-list>
<servlet>
  <servlet-name>Login</servlet-name>
  <servlet-class>Login</servlet-class>

</servlet>
<servlet-mapping>
  <servlet-name>Login</servlet-name>
  <url-pattern>/Login</url-pattern>
</servlet-mapping>
```

앞서 만들었던 Login.java 서블렛과 main.jsp를 web.xml에서 mapping시켜줍니다.
이로써 이 Login.java 서블렛은 main.jsp에서 작동하게 됩니다.

Change.java(1)

```
<form id=form2  method="POST">


    <li class="nav-item">
        <h4 class="nav-link">비밀번호를 입력하세요.</h4>
    </li>
    <li class="nav-item">
        <input class ="nav-link" type="password" id="pw">
    </li>
    <li>
        <input class ="nav-link" type="button" onclick ="adminInfo()"value="정보확인">
    </li>

</form>
```

```
function adminInfo(){

    var pw = document.getElementById("pw").value;
    var form = document.getElementById("form2");
    form.setAttribute("action", "./Change?pw="+pw+"");
    form.submit();//
    alert("과연결과는?");


}
```

admin.jsp의 form에 대해서 받은 값들을 submit 해주어 Change.java로 값을 넘겨줍니다.

Change.java(2)

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    // TODO Auto-generated method stub

    HttpSession session = request.getSession();
    String pw = request.getParameter("pw");

    session.setAttribute("pw", pw);

    // 관리자 계정으로 로그인
    if(pw.equals("12345"))
    {
        String admin = "adminChange.jsp";
        request.getRequestDispatcher(admin).forward(request, response);
    }
    // 일반 계정으로 로그인
    else
    {
        String login = "main.jsp";
        request.getRequestDispatcher(login).forward(request, response);
    }
}
```

Admin.jsp에서 form에 쓰인 metho방식대로 값을 받아 doGet이 호출됩니다.
이 서블렛은 입력한 password가 요구하는 password가 맞는지 판단합니다.
getRequestDispatcher()를 통해서 처리 속도를 향상시켰습니다.

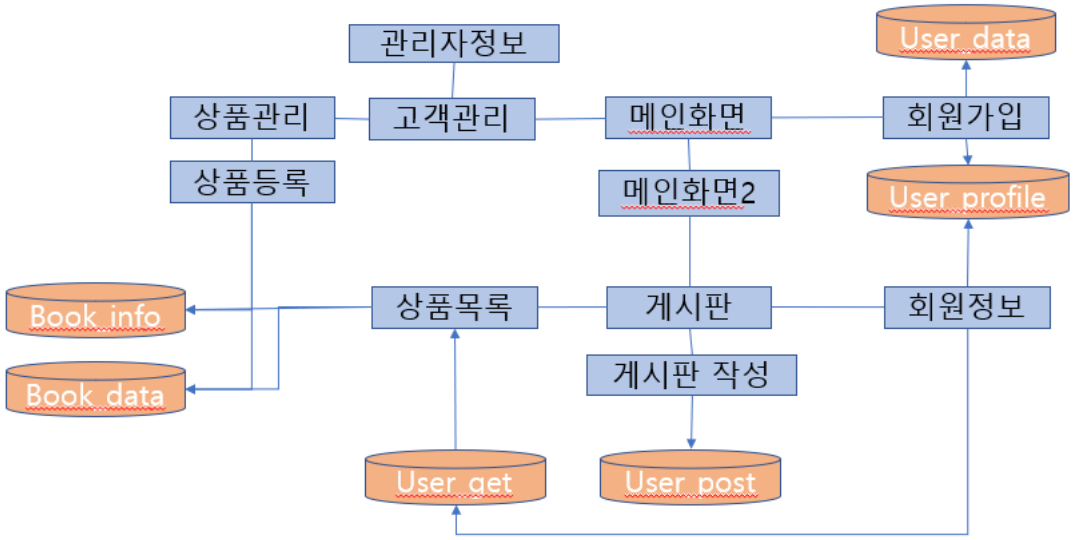
Change.java(3)

```
<welcome-file-list>
  <welcome-file>admin.jsp</welcome-file>
</welcome-file-list>
<servlet>
  <servlet-name>Change</servlet-name>
  <servlet-class>Change</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Change</servlet-name>
  <url-pattern>/Change</url-pattern>
</servlet-mapping>
```

앞서 만들었던 Change.java 서블렛과 admin.jsp를 web.xml에서 mapping시켜줍니다.
이로써 이 Change.java 서블렛은 admin.jsp에서 작동하게 됩니다.

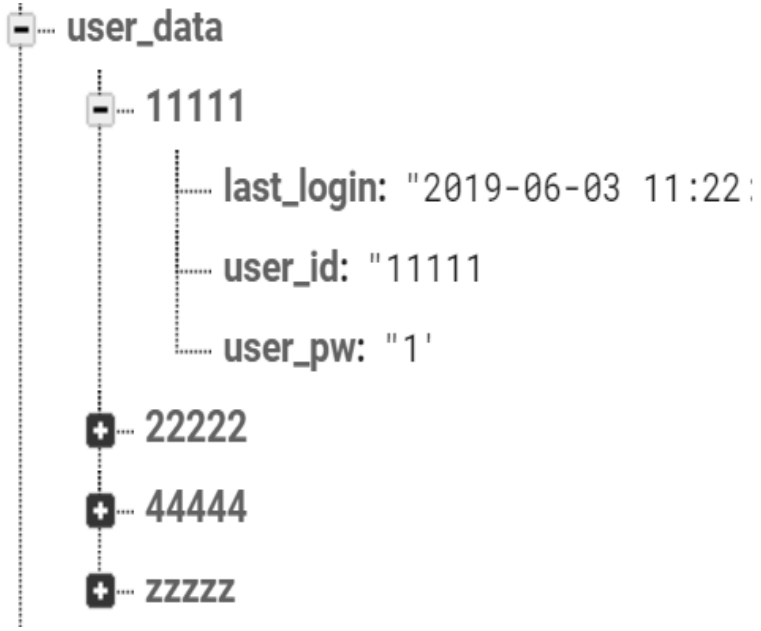
6. 데이터베이스 구조

데이터베이스 구성



데이터 베이스는 총 6개로 구성되었습니다.

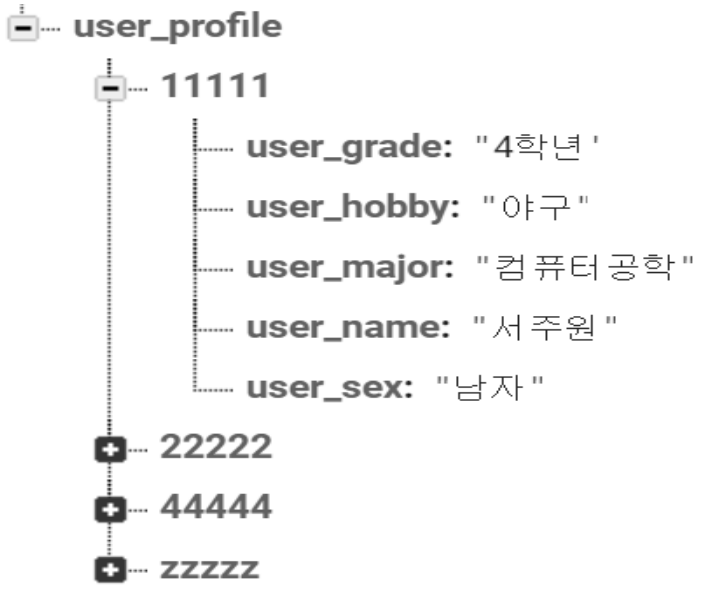
user_data 테이블



- User_data table

사용자의 ID, PW, 마지막 로그인 시간을 저장합니다.

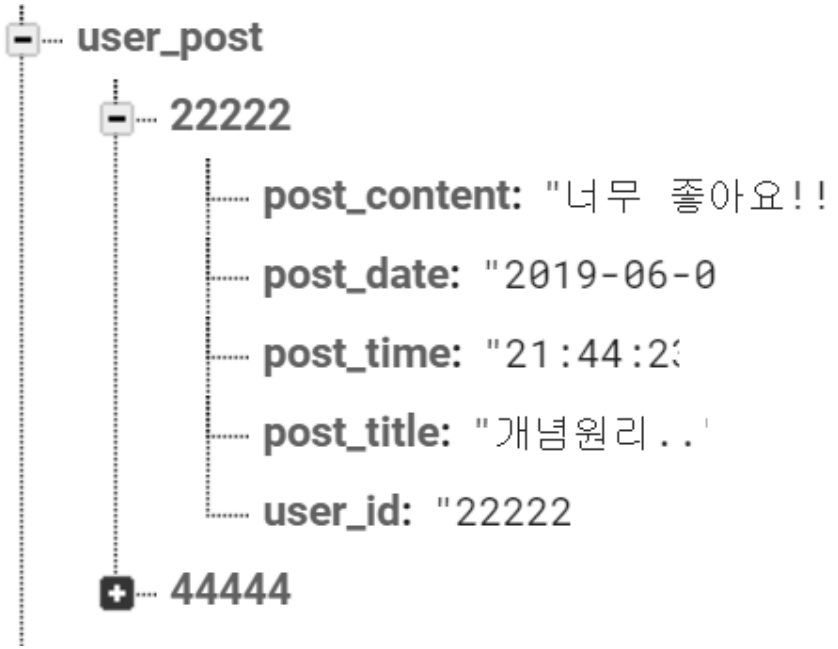
user_profile 테이블



- User_profile table

사용자의 아이디 별로 관리하며,
하위에 이름, 성별, 학년, 전공, 취미를
기록합니다.

User_post 테이블

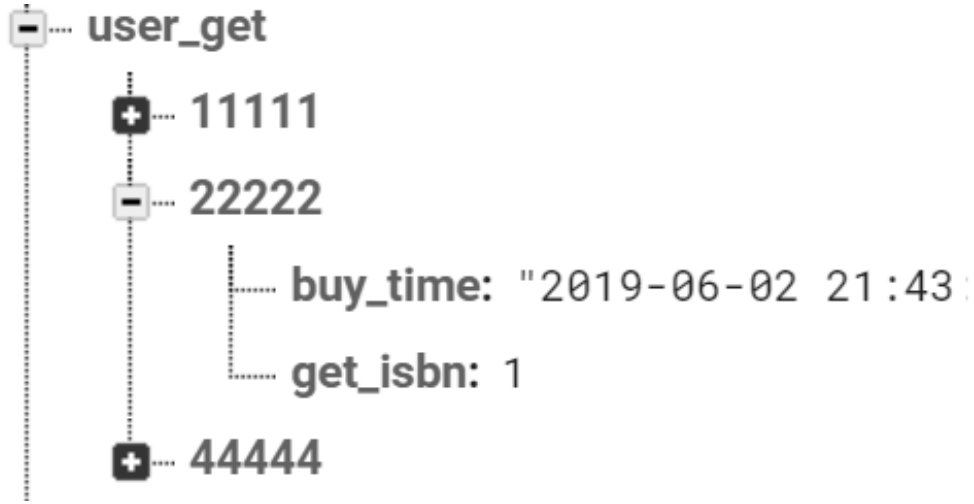


- User_post table

사용자의 아이디 별로 관리하며,

하위에 사용자의 아이디, 포스트 제목, 포스트 내용, 작성 날짜, 작성 시간을 저장합니다.

user_get 테이블

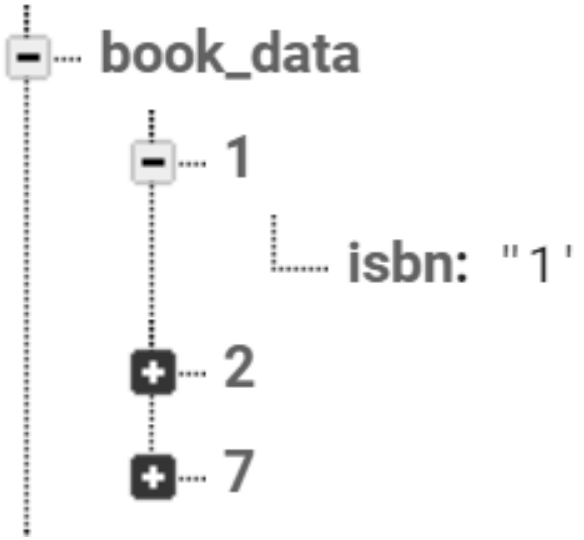


- User_get table

사용자의 ID별로 관리되며,
구매한 책의 isbn번호와 구매 시간을 저장합니다.

Isbn만 저장시켜서 해당 값으로
Book_info table과 join을 통해 값을 찾기 위해서 입니다.

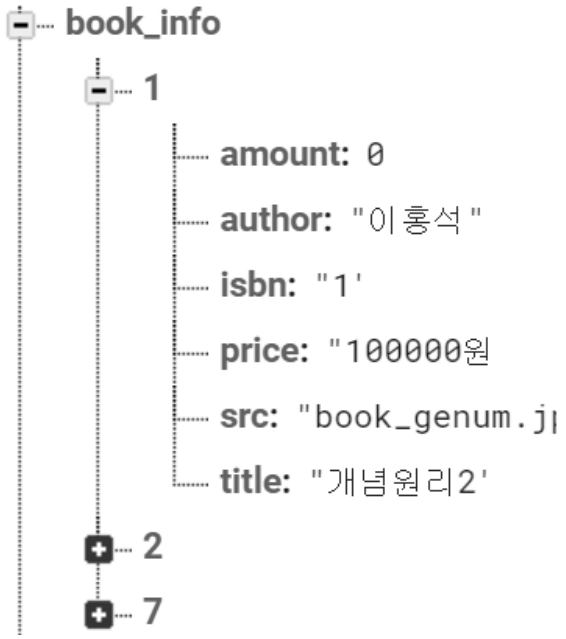
book_data 테이블



- book_data table

책의 isbn에 대해서 저장합니다.

book_info 테이블



- book_info table

책의 isbn 별로 관리하며,

하위에 제목, 가격, 작가, 이미지 파일, 개수를 기록합니다.

7. 데이터베이스 연산

DB insert연산

```
var user_data = firebase.database().ref('user_data/'+id);
user_data.set({
    user_id: id,
    user_pw: pw,
    last_login: getTimeStamp()
});
```

.set을 사용해서

필요로 하는 값을 DB에 추가시켜줍니다.

DB update연산(1)

```
var edit = firebase.database().ref('user_profile/' + '<%= id %>');
edit.update
({
    user_name: _name,
    user_sex: _sex,
    user_major: _major,
    user_grade: _grade,
    user_hobby: _hobby
});
```

.update을 사용해서 사용자가 edit 값들을
기존에 있던 data에 update 시켜줍니다.

DB update연산(2)

```
var edit_book_info = firebase.database().ref('book_info/' + isbn);
edit_book_info.once('value', function(snapshot) {
var tmp = snapshot.val();

var _title = tmp.title;
var _author = tmp.author;
var _price = tmp.price;
var _src = tmp.src;
var _amount = tmp.amount;

edit_book_info.update({

    title : _title,
    author : _author,
    price : _price,
    src : _src,
    amount : _amount-1
});
```

.update을 사용해서 사용자가 책을 구매하면
해당 책의 info 값 중 amount값을 1빼서
Update 시켜줍니다.

DB delete 연산

```
function removeAccount()
{
        var ret = confirm("정말 탈퇴하시겠습니까?");
        if(ret)
        {
            var rmv_data = firebase.database().ref('user_data/' + '<%= id %>');
            var rmv_profile = firebase.database().ref('user_profile/' + '<%= id %>');
            var rmv_post = firebase.database().ref('user_post/' + '<%= id %>');
            var rmv_get = firebase.database().ref('user_get/' + <%= id %>);

            rmv_data.remove();
            rmv_profile.remove();
            rmv_post.remove();
            rmv_get.remove();

            alert("탈퇴되었습니다");
            location.href = "main.jsp";
        }
        else
            alert("탈퇴가 취소되었습니다");
}
```

유저가 회원탈퇴를 눌렀을 때, .remove()를 이용해서 해당 유저의 id를 키로 가는지 값을 여러 DB에서 삭제한다.

감사합니다.