

ECO-FRIENDLY SERVICE PROJECT



지구를 위한 작은 발걸음
ECO-FRIENDLY SERVICE PROJECT

작성자 : 한석희

만든기간 : 2024.05.02 ~ 2024.05.14



01 프로젝트 주제

친환경 자전거 대여 서비스에 대한 업그레이드 서비스 개발

02 ECO 서비스의 선정 배경

요즘 현대인들의 대중적인 교통문화로 발달된 전기 자전거 및 전동 킥보드에 대해 사용자에게 더 편리한 방법을 제공 할 수 있는 방법을 모색하게 되어 예약을 하면 원하는 위치에 배송을 할 수 있는 시스템을 개발하고자 함

03 기획 의도

바쁜 현대 사회에 대한 빠르고 편리한 이동수단인 자전거에 대해 전기 자전거 및 킥보드 대여시 대여 위치에 대한 불편함을 최소화 하기 위한 온라인 예약 및 배송 서비스로 사용자에게 불편함 요소를 제거하기 위함

04 프로젝트 내부 주요 코드

로그인/로그아웃 시스템, 장바구니, 게시판 등 사용자에게 편의 기능에 대한 코드 개발

05 프로젝트에 대한 개선 방향

제작하게 되면서 부족한 부분에 대한 느낀점

06 마무리하며...

전체적인 프로젝트에 대한 느낀점

MAIN CODE



첫째, Join Membership & Login/Logout System

회원가입 시 DB에 회원 정보에 대한 데이터를 저장.

로그인 시, DB에 해당 아이디와 비밀번호 확인 후
일치시에만 세션 및 쿠키 생성
로그아웃 시, 세션 및 쿠키 삭제



둘째, Shopping Basket System

페이지 이동시마다 해당 계정의 ID의 쿠키와 세션이 일치
하는지 확인.

상품의 개수를 선택 후 장바구니 담기 클릭 시 해당 상품의
Name과 Value값을 쿠키에 저장하여 구매시에 장바구니
에 담은 데이터만 추출하여 구매



셋째, Bulletin Board System

Thymeleaf를 이용하여 DB에 저장되어 있는 게시물
의 데이터 만큼 게시물 페이지에 표시.

Controller에서 PathVariable을 이용하여 1개의
페이지를 이용하여 원하는 게시물의 DB로 이동하는
코드 구성

01 프로젝트 주제

[온라인 자전거 대여 서비스 홈페이지 개발]

다수의 사용자가 전기 자전거 및 전동 킥보드를 이용하는 상황에서, 더 편리하고 다양한 자전거 및 전동 킥보드를 대여할 수 있는 온라인 자전거 대여 서비스 웹사이트를 개발하게 되었습니다.

02 ECO 서비스의 선정 배경

[환경 보호와 편리한 이동을 위한 Eco 서비스 개발 배경]

현대 사회에서 전기 자전거 및 전동 킥보드의 보급이 늘어나면서 대여 서비스의 수요가 높아지고 있는 추세입니다. 이러한 대여 서비스의 인기를 실감하여 프로젝트로서 대여 서비스 홈페이지를 개발 해보게 되었습니다.

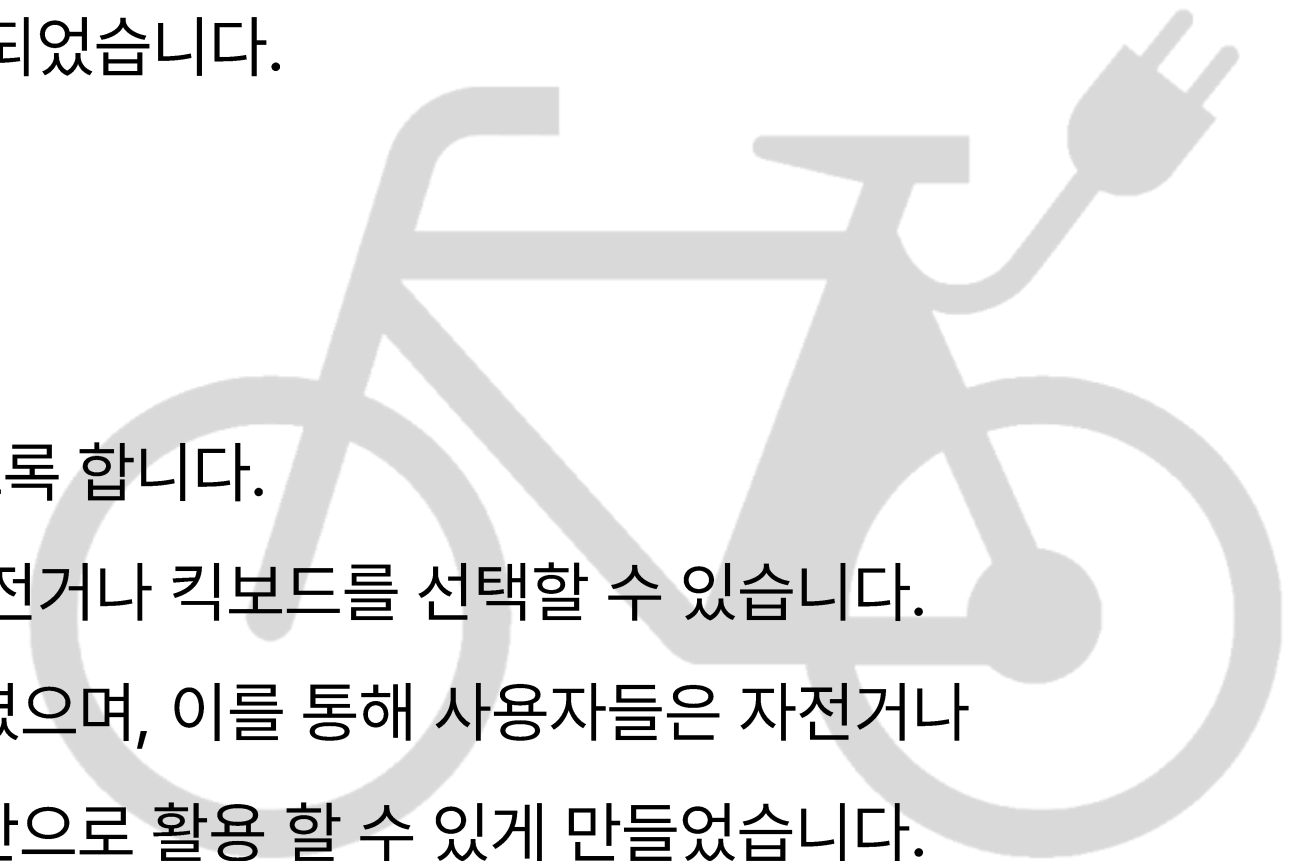
03 기획의도

[온라인 대여부터 배송까지]

사용자들이 편리하게 다양한 종류의 전기 자전거 및 전동 킥보드를 대여하고 사용할 수 있도록 합니다.

이 서비스를 통하여 사용자들은 컴퓨터를 통해 간편하게 대여 예약을 할 수 있고, 필요한 자전거나 킥보드를 선택할 수 있습니다.


또한, 배송 서비스를 제공하여 사용자가 원하는 장소로 편리하게 배달 받을 수 있게 기획하였으며, 이를 통해 사용자들은 자전거나 킥보드를 이 전 처럼 대여할 위치를 찾지 않아도 필요할 때마다 편리하게 대여하여 이동 수단으로 활용 할 수 있게 만들었습니다.



04

프로젝트 내부 주요 코드 - ① Join Membership & Login/Logout System

로그인



또는 이메일을 사용하여 로그인 하세요

ID or Email

Password

비밀번호를 잊으셨나요?

SIGN IN

SIGN UP

어서오세요!

편리한 ECO 자전거 대여를 위해 로그인을 해주세요.

SIGN IN

회원가입

ID

Name

Password

re-enter your password

Main Address

Search

Sub Address


Postal Code

Secure_Num1

Secure_Num2

SIGN UP

비밀번호 찾기



아이디 또는 이메일을 입력해 주세요.

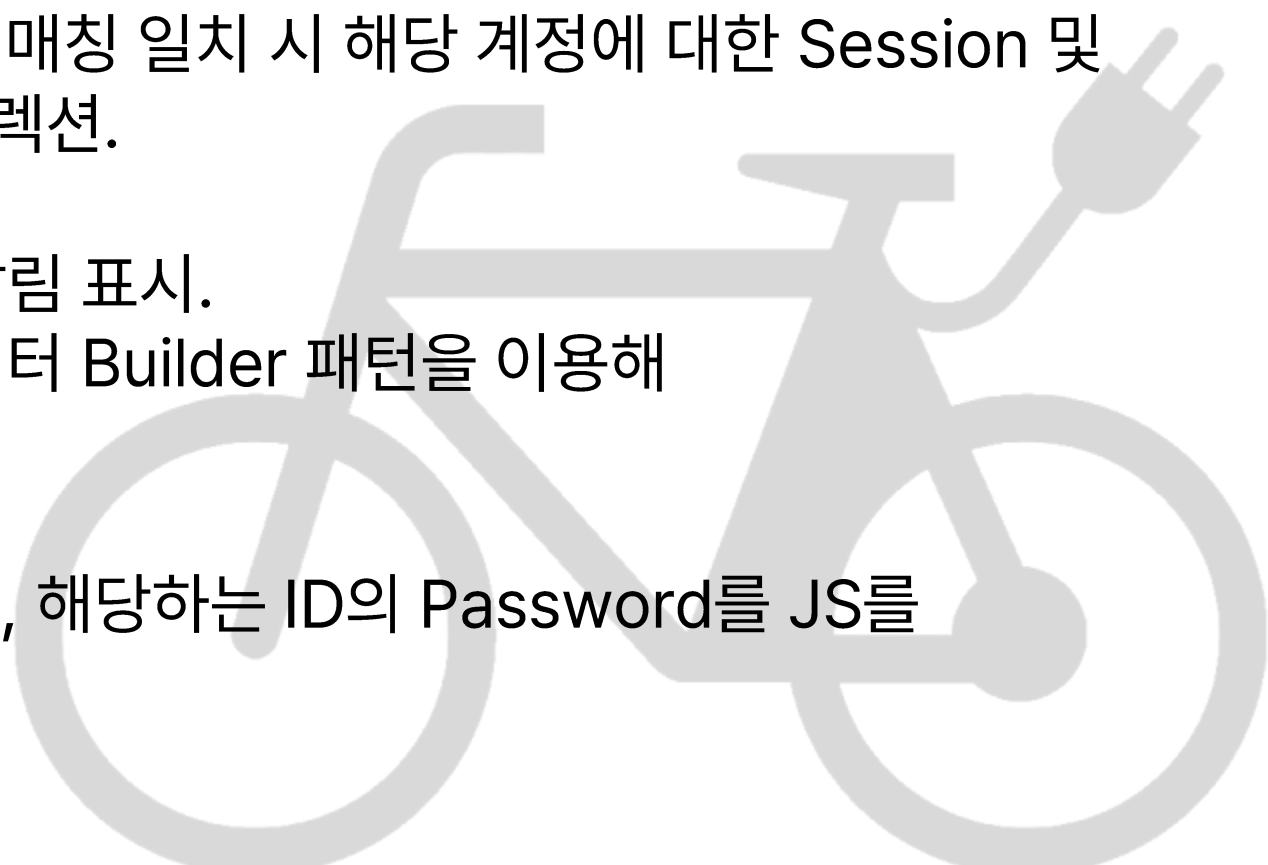
ID or Email

Find

SIGN IN

{ Sign In[로그인], Sign Up[회원가입], Find Password[비밀번호찾기] }

1. Sign In : INFO DB에 저장되어 있는 데이터를 통하여 사용자가 입력한 ID & Password를 매칭 일치 시 해당 계정에 대한 Session 및 Cookie 생성. 만약 없을 시, JS를 통한 오류 알림창 표시 및 로그인 페이지 리다이렉션.
2. Sign Up : 신규 가입자에게 정규 표현식 맞추어 회원가입 요청. 정규표현식 틀릴 시 오류 알림 표시.
Daum 주소 API를 이용한 주소찾기 시스템 구축하여 Sign UP 클릭시 해당 데이터 Builder 패턴을 이용해 INFO DB에 저장. 이 후 Sign In 페이지로 이동.
3. Find Password : INFO DB에 저장되어 있는 데이터와 사용자가 입력한 ID를 매칭시킨 뒤, 해당하는 ID의 Password를 JS를 이용하여 알림창을 통해 표시.



04

프로젝트 내부 주요 코드 - ① Join Membership & Login/Logout System

```

@PostMapping("/login_home")
public String login_home_Post(@RequestParam(name = "id") String id,
                              @RequestParam(name = "password") String pw,
                              Model mo,
                              HttpServletRequest request,
                              HttpServletResponse response) {

    String user_id = Ma.UserId(id);
    ① String user_pw = Ma.UserPw(id);

    ② if(id.equals(user_id) && pw.equals(user_pw)){
        if(id.equals("ADMIN") || id.equals("admin")){
            HttpSession session = request.getSession();
            session.setAttribute("ADMIN", id);
            Cookie ID_Cookie = new Cookie("ADMIN", id);
            ID_Cookie.setMaxAge(10000);
            response.addCookie(ID_Cookie);
            mo.addAttribute("Display", Ma.DisplaySet().get(0));
            mo.addAttribute("info", id);
            return "/Admin/Admin_Page.html";
        }
        mo.addAttribute("info", user_id);
        HttpSession session = request.getSession();
        session.setAttribute("ID", id);
        Cookie ID_Cookie = new Cookie("ID", id);
        ID_Cookie.setMaxAge(10000);
        response.addCookie(ID_Cookie);
        return "/Home/main.html";
    }
    mo.addAttribute("error", "아이디 또는 비밀번호가 올바르지 않습니다.");
    ③ return "/Sign/SignIn_Page.html";
}

```

{ Sign In -> Main Page 이동시 Controller }

1. user_id와 user_pw를 사용자가 입력한 id를 이용해 DB에서 가져온다.
2. DB에서 id, pw를 확인한 뒤, 관리자와 일반사용자를 나누어 세션과 쿠키를 생성한다. 관리자는 관리자 페이지로 이동하고, 일반사용자는 Main Page로 이동한다.
3. id 또는 pw가 틀릴 경우, Model을 사용하여 오류 메시지를 HTML로 전송한다.



04

프로젝트 내부 주요 코드 - ① Join Membership & Login/Logout System

```

@PostMapping("/SignIn_Page")
public String signupin(@RequestParam(required=false) String Id,
                      @RequestParam(required=false) String Name,
                      @RequestParam(required=false) String Pw1,
                      @RequestParam(required=false) String Maddr,
                      @RequestParam(required=false) String Saddr,
                      @RequestParam(required=false) String Paddr,
                      @RequestParam(required=false) String S1,
                      @RequestParam(required=false) String S2) {

```

```

① UpDB info_up = new UpBuild().
    ID(Id).NAME(Name).
    PW(Pw1).MADDR(Maddr).
    SADDR(Saddr).
    PADDR(Paddr).
    S1(S1).|
    S2(S2).
    Biluder();

```

```

② Ma.Insert(info_up.getId(),
            info_up.getName(),
            info_up.getPw(),
            info_up.getMaddr(),
            info_up.getSaddr(),
            info_up.getPaddr(),
            info_up.getS1(),
            info_up.getS2());
return "Sign/SignIn_Page.html";
}

```

{ Sign Up -> Sign In 이동시 Controller }

1. 사용자가 입력한 회원정보를 회원정보 Builder 패턴을 이용해 저장한다.
2. Builder 패턴으로 저장한 회원정보를 DB에 저장한다.



04 프로젝트 내부 주요 코드 - ① Join Membership & Login/Logout System

```
@GetMapping("/")
public String home(HttpSession session,
    HttpServletResponse response,
    HttpServletRequest request,
    Model mo) {
    ① CookieUtil cookieUtil = new CookieUtil();
    Cookie[] cookies = request.getCookies();
    String checkUser = cookieUtil.matchUser(cookies);
    mo.addAttribute("info", checkUser);
    return cookieUtil.matchSession(request, response, cookies, "/Home/main.html");
}

@GetMapping("/logout")
public String logout(HttpSession session,
    HttpServletResponse response,
    HttpServletRequest request) {
    ② session.invalidate();
    Cookie[] cookies = request.getCookies();
    for (Cookie cookie : cookies) {
        if(cookie == null){
            return "redirect:/";
        }
        cookie.setMaxAge(0);
        response.addCookie(cookie);
    }
    return "redirect:/";
}

@GetMapping("/login_home")
public String login_home2(HttpServletRequest request,
    HttpServletResponse response,
    Model mo) {
    ① CookieUtil cookieUtil = new CookieUtil();
    Cookie[] cookies = request.getCookies();
    String checkUser = cookieUtil.matchUser(cookies);
    mo.addAttribute("info", checkUser);
    return cookieUtil.matchSession(request, response, cookies, "/Home/main.html");
}
```

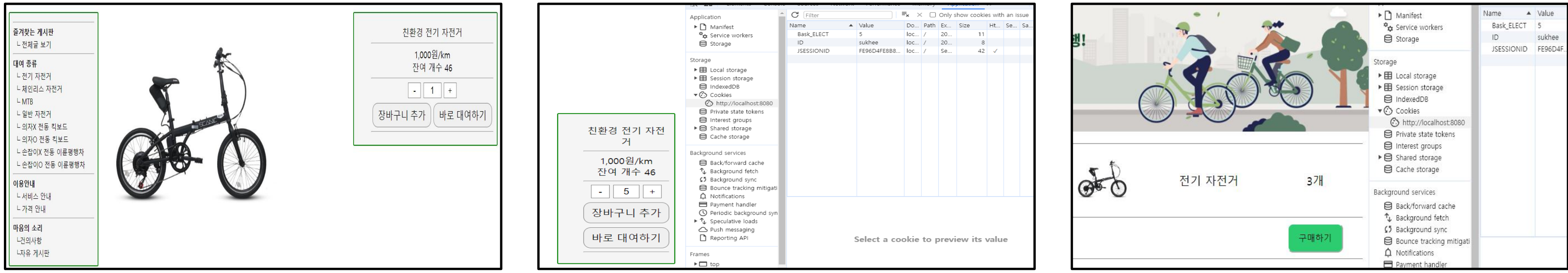
```
<ul class="lo_ul" th:if="${info == null}">
    <li><a href="/SignIn">sign-in</a></li>
    <li><a href="/SignUp">sign-up</a></li>
    <li onclick="login()" style="cursor:pointer;">service</li>
</ul>
<ul class="lo_ul" th:if="${info != null}">
    <li>[[${info}]]님 환영합니다.</li>
    <li><a href="/Logout">logout</a></li>
    <li><a href="/service">service</a></li>
</ul>
```

{ Main Page Controller }

- 1. [CookieUtil](#) 클래스를 이용하여 쿠키의 데이터를 이용하여 데이터의 존재 여부를 확인하고, 있을 시에는 이동하고자 하는 페이지로 이동한다.
- 2. Main Page에서 Logout 버튼을 클릭했을 때는 세션과 사용자가 가지고 있는 Cookie 데이터를 모두 삭제하고, "/"경로로 이동하여 Main Page로 다시 이동시킨다.



04 프로젝트 내부 주요 코드 - ② Shopping Basket System



{ Product[상품 페이지], Basket[장바구니], Buy[구매 페이지] }

- 1.Product : 상품별 페이지를 생성하여 Cookie와 Session이 있을 시에만 상품 페이지로 이동 가능.
이동 후 관리자가 설정한 제품의 수량이 ADMIN_TB DB에 저장되어 있는 데이터를 가져와 상품별 페이지에 현재 잔여 수량을 표시.
- 2. Basket : HTML에서의 JS를 통한 INPUT value의 값을 "+"와 "-"를 이용하여 변경하고, 장바구니 추가를 클릭시에는 해당하는 값이 Cookie 데이터에 제품의 간략한 이름과 value값에 사용자가 입력한 값만큼 추가 된다. 해당 Cookie 데이터는 다른 상품 페이지를 가도 유지가 되며, 로그아웃 또는 구매를 해야지만 해당 쿠키가 제거된다.
- 3. Buy : 바로 대여하기를 클릭해야 이동하는 페이지이며, 장바구니에 추가한 Cookie 데이터와 바로 대여하기를 한 상품 페이지에 value값만 Buy Page에 나타나게 되며, 만약 장바구니와 바로 대여하기의 상품이 겹치게 되면 바로 대여하기가 우선순위로 Buy페이지에는 장바구니의 Cookie value값이 나오지 않고 바로 대여하기한 상품 개수가 표시된다. Buy Page에서 구매하기를 클릭했을 시 해당 데이터는 ADMIN_TB DB에 수량에서 구매한 value의 값을 차감하여 ADMIN_TB DB에 저장한뒤, 모든 상품별 페이지에 반영하게 된다.

04 프로젝트 내부 주요 코드 - ② Shopping Basket System

```
@PostMapping("/findPW")
public String findPW(@RequestParam(required=false, name="id") String id,
                    Model mo)
{
    ① if(Ma.UserId(id) != null)
    {
        String findPW = Ma.UserPw(id);

        mo.addAttribute("findPW", "해당하는 비밀번호는 : "+findPW);

        return "/Sign/SignIn_Page.html";
    }

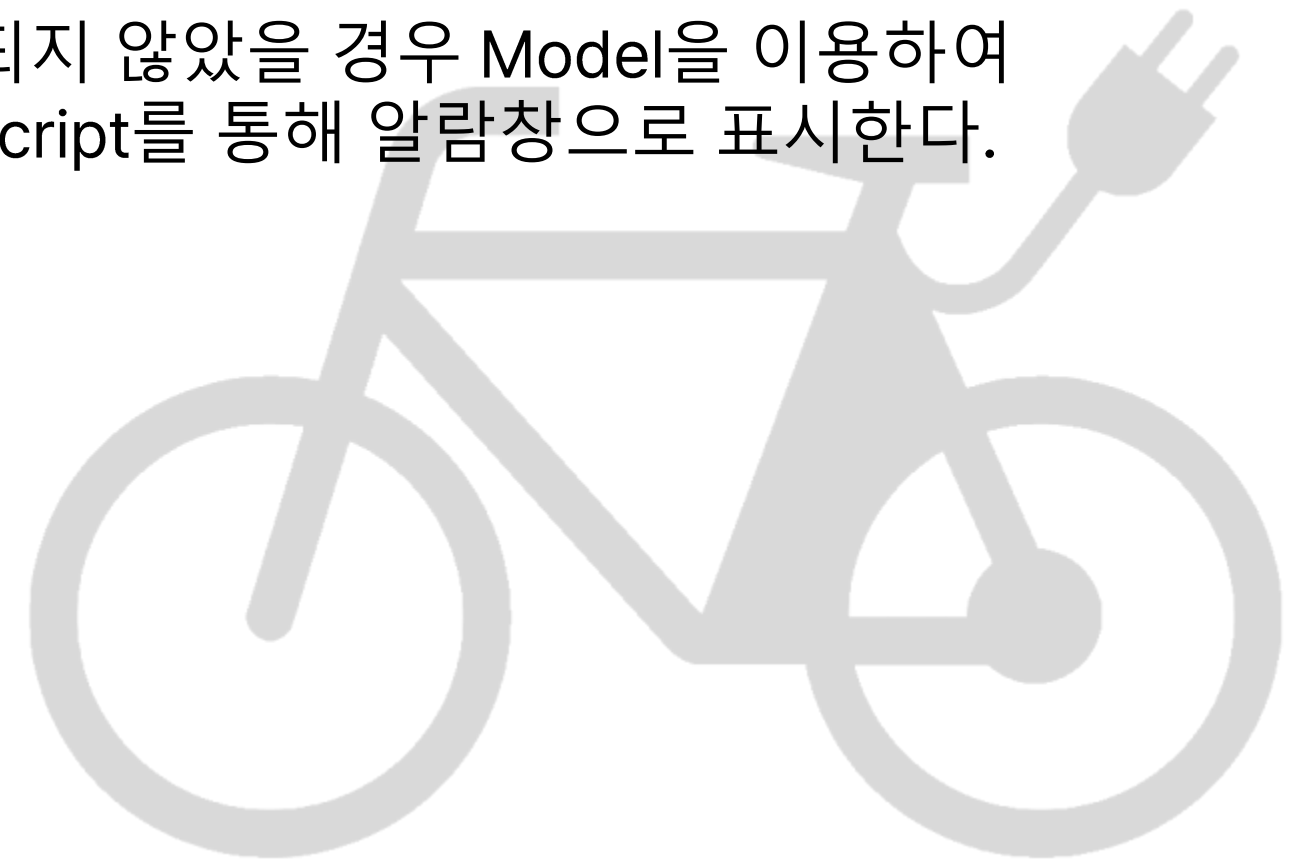
    ② mo.addAttribute("error", "해당하신 아이디는 존재하지가 않습니다.");

    return "/Sign/forgetPW.html";
}
```

{ Main Page -> Product Page 이동시 Controller }

1. @Autowired 어노테이션을 사용하여 Mapper를 연결한 후, 사용자가 입력한 회원정보를 @RequestParam 어노테이션을 이용하여 사용자가 입력한 id 데이터를 HTML에서 가져온다. 그 다음, Mapper의 DB안에 있는 INFO 테이블에 사용자가 입력한 해당 id의 값이 있는지 확인한 뒤, 있으면 Model을 이용하여 HTML로 사용자가 입력한 id의 findPW를 반환한다.

2. IF문에서의 조건이 충족되지 않았을 경우 Model을 이용하여 error값을 반환한 뒤, JavaScript를 통해 알람창으로 표시한다.



04 프로젝트 내부 주요 코드 - ② Shopping Basket System

```

@GetMapping("/elect_basket")
public String electbasket(@RequestParam(name="count") int count,
                          HttpServletRequest request,
                          HttpServletResponse response,
                          Model mo) {
    ① CookieUtile cookieUtile = new CookieUtile();
    Cookie[] cookies = request.getCookies();
    String checkUser = cookieUtile.matchUser(cookies);

    ② mo.addAttribute("info", checkUser);
    mo.addAttribute("Display", Ma.DisplaySet().get(0));

    ③ String cookieName = "Bask_ELECT";
    int basketCount = cookieUtile.getBasketCount(request.getCookies(), cookieName);
    Cookie ID_Cookie = new Cookie(cookieName, String.valueOf(count + basketCount));

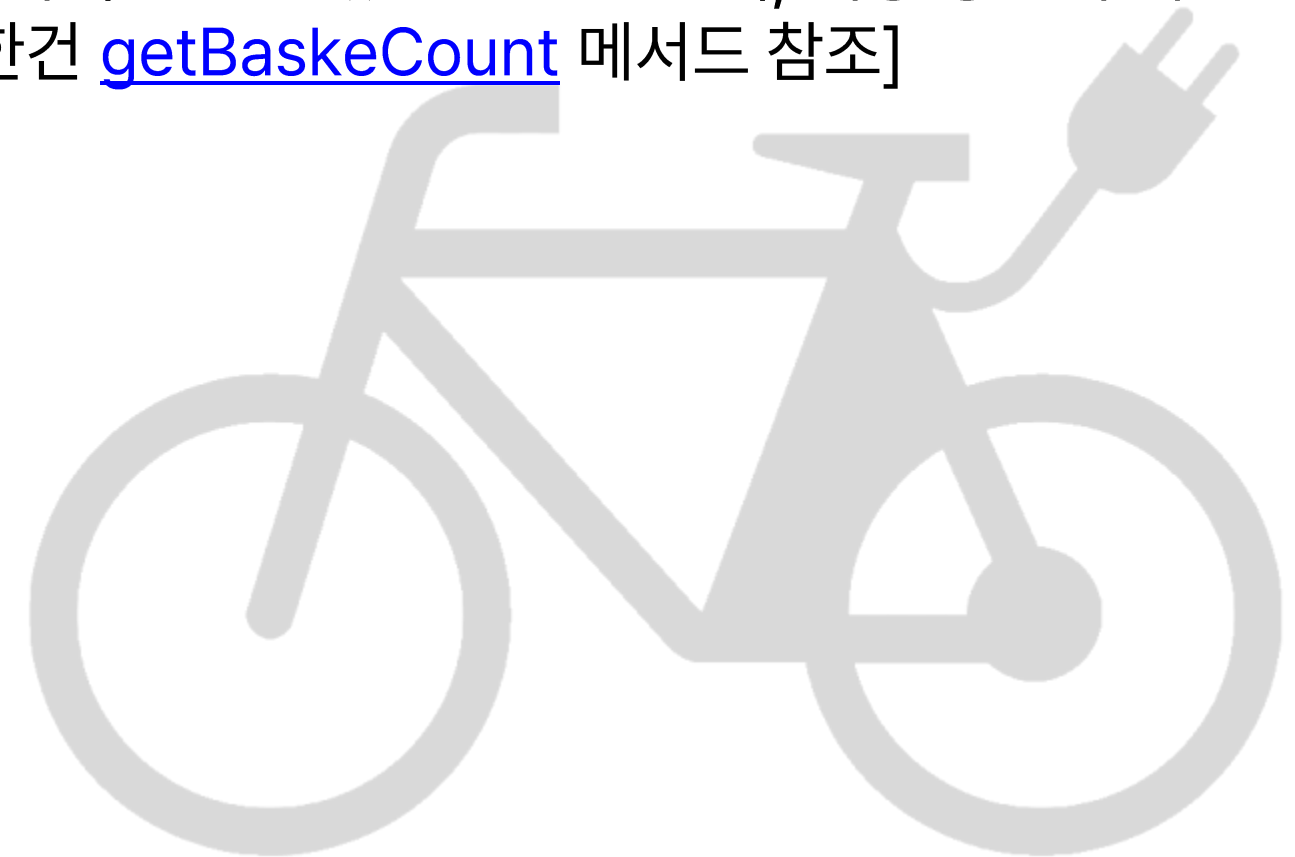
    ID_Cookie.setMaxAge(10000);
    response.addCookie(ID_Cookie);

    return cookieUtile.matchSession(request, response, cookies, "/Product/Electro.html");
}

```

{ Basket Action Controller }

1. [CookieUtile](#) 클래스를 이용하여 쿠키의 데이터를 이용하여 데이터의 존재 여부를 확인하고, 있을 시에는 이동하고자 하는 페이지로 이동한다.
2. ①에서 Cookie의 사용자 id를 담은 checkUser를 생성한뒤, 제품의 개수와 함께 HTML로 데이터를 전송한다.
3. 상품에 대한 cookieName을 정해준 뒤, CookieUtile 내의 장바구니 쿠키의 개수를 파악하는 메서드를 이용하여 Value 값을 반환 받은 뒤, 해당 상품에 대한 새로운 쿠키를 생성한다.[자세한건 [getBaskeCount](#) 메서드 참조]



04 프로젝트 내부 주요 코드 - ② Shopping Basket System

```
@GetMapping("/shopping")
public String shopping(@RequestParam(name="count") int count,
                      @RequestParam(name="productName") String productName,
                      HttpServletRequest request, HttpServletResponse response,
                      Model mo) {
    ① CookieUtil user = new CookieUtil();
    Cookie[] cookies = request.getCookies();
    String checkUser = User.matchUser(cookies);

    ② List<String> All_Data = new ArrayList<>();
    List<String> Data_img = new ArrayList<>();

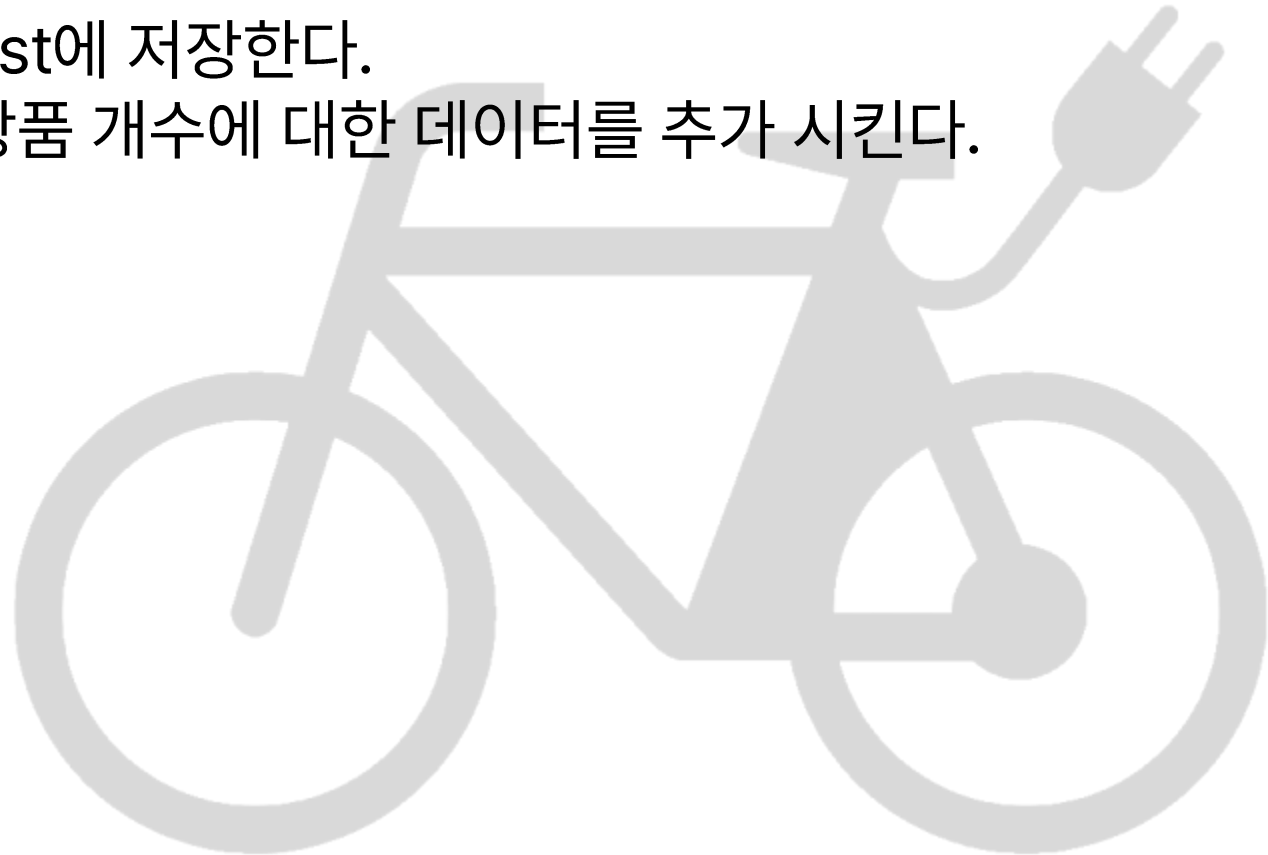
    ③ int Elect_value = 0;
    int Chain_value = 0;
    int MTB_value = 0;
    int Nomal_value = 0;
    int Nokick_value = 0;
    int Onkick_value = 0;
    int Nocar_value = 0;
    int Oncar_value = 0;

    ④ switch (productName) {
        case "chain":
            Chain_value += count;
            All_Data.add("체인리스 자전거");
            Data_img.add("체인리스 자전거.jpg");
            break;
        case "electro":
            Elect_value += count;
            All_Data.add("전기 자전거");
            Data_img.add("전기 자전거.jpg");
            break;
        case "mtb":
            MTB_value += count;
            All_Data.add("MTB");
            Data_img.add("MTB.jpg");
            break;
    }
```

```
<form id="basketForm" action="/elect_basket" method="get">
    <input type="hidden" id="basketCount" value="1" name="count">
</form>
<form id="shoppingForm" action="/shopping" method="get">
    <input type="hidden" id="shoppingCount" value="1" name="count">
    <input type="hidden" id="productName" name="productName" value="electro">
</form>
```

{ Buy Action Controller - 1 }

- 1. [CookieUtil](#) 클래스를 이용하여 쿠키의 데이터를 이용하여 데이터의 존재 여부를 확인하고, 있을 시에는 이동하고자 하는 페이지로 이동한다.
- 2. 구매시 상품 표시에 대한 데이터를 Data와 IMG로 나누어 List에 저장한다.
- 3. 상품을 바로 대여하기 했을때의 개수를 저장하기 위한 공간을 생성한다.
- 4. HTML에서 해당 상품의 value 값으로 switch case를 통해 바로 대여하기의 개수를 각 데이터 List에 저장한다.
※swutch case문이 끝난 뒤, 상품 개수에 대한 데이터를 추가 시킨다.



04 프로젝트 내부 주요 코드 - ② Shopping Basket System

```
@PostMapping("/Thanks")
public String payToThanks(HttpServletRequest request,
                          HttpServletResponse response,
                          Model mo,
                          @RequestParam("productName") List<String> productNames,
                          @RequestParam("productOrderCount") List<Integer> productOrderCounts) {
    ❶ CookieUtil user = new CookieUtil();
    Cookie[] cookies = request.getCookies();
    String checkUser = user.matchUser(cookies);
    mo.addAttribute("info", checkUser);
    AdminDB productInfo = Ma.DisplaySet().get(0);

    ❷ List<ProductOrder> orders = new ArrayList<>();

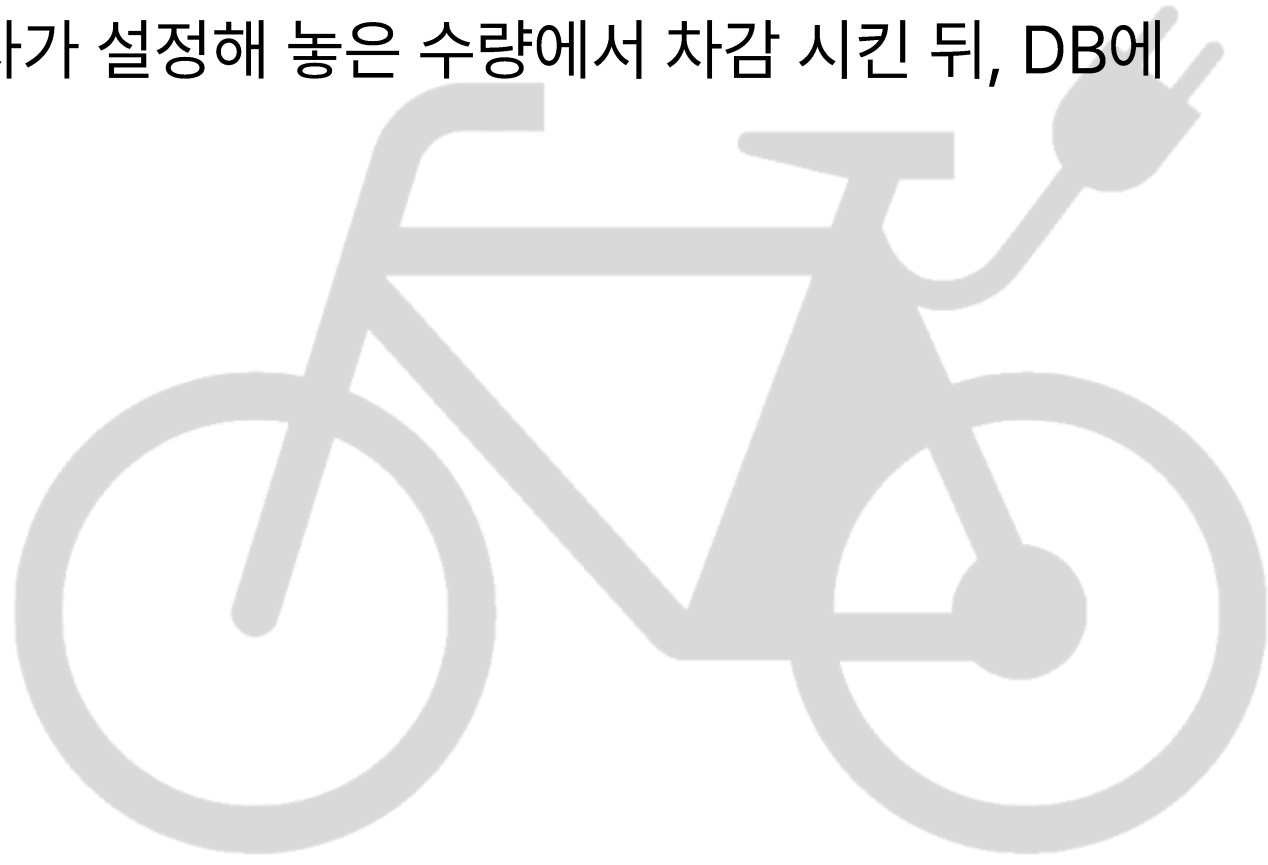
    ❸ for (int i = 0; i < productNames.size(); i++) {
        orders.add(new ProductOrder(productNames.get(i), productOrderCounts.get(i)));
    }

    ❹ for (ProductOrder order : orders) {
        if ("전기 자전거".equals(order.getProductName())) {
            Ma.updateElect(String.valueOf(Integer.parseInt(productInfo.getElect()) - order.getProductOrderCount()));
        } else if ("체인리스 전기 자전거".equals(order.getProductName())) {
            Ma.updateChain(String.valueOf(Integer.parseInt(productInfo.getChain()) - order.getProductOrderCount()));
        } else if ("MTB".equals(order.getProductName())) {
            Ma.updateMtb(String.valueOf(Integer.parseInt(productInfo.getMtb()) - order.getProductOrderCount()));
        } else if ("일반 자전거".equals(order.getProductName())) {
            Ma.updateNomal(String.valueOf(Integer.parseInt(productInfo.getNomal()) - order.getProductOrderCount()));
        } else if ("전동 킥보드[의자 X]".equals(order.getProductName())) {
            Ma.updateNokick(String.valueOf(Integer.parseInt(productInfo.getNokick()) - order.getProductOrderCount()));
        } else if ("전동 킥보드[의자 O]".equals(order.getProductName())) {
            Ma.updateOnkick(String.valueOf(Integer.parseInt(productInfo.getOnkick()) - order.getProductOrderCount()));
        } else if ("전동 이륜평행차[손잡이 X]".equals(order.getProductName())) {
            Ma.updateNocar(String.valueOf(Integer.parseInt(productInfo.getNocar()) - order.getProductOrderCount()));
        } else if ("전동 이륜평행차[손잡이 O]".equals(order.getProductName())) {
            Ma.updateOncar(String.valueOf(Integer.parseInt(productInfo.getOncar()) - order.getProductOrderCount()));
        }
    }

    return user.matchSession(request, response, cookies, "/Product/Thanks.html");
}
```

{ Buy Action Controller - 2}

- 1. [CookieUtil](#) 클래스를 이용하여 쿠키의 데이터를 이용하여 데이터의 존재 여부를 확인하고, 있을 시에는 이동하고자 하는 페이지로 이동한다.
- 2. [ProductOrder](#) 클래스를 이용하여 최종적으로 구매할 개수에 대한 데이터를 저장할 공간을 생성한다.
- 3. [HTML](#)에서 데이터를 받아온뒤, List 배열에 저장시킨다.
- 4. 각 상품에 대한 개수를 관리자가 설정해 놓은 수량에서 차감 시킨 뒤, DB에 다시 저장한다.



04

프로젝트 내부 주요 코드 - ③ Bulletin Board System

▷ 총 3개의 게시물이 있습니다. SELECT 검색

번호	제목	글쓴이	일시
28	sukhee	안녕하세요	2024-05-11 00:00:00
30	sukhee	관리자입니다.	2024-05-13 00:00:00
31	sukhee	가입인사드립니다.	2024-05-13 00:00:00
32	admin	제가 관리자입니다.	2024-05-13 00:00:00
33	admin	자전거와 함께하는 즐거운 여행!	2024-05-13 00:00:00

작성하기

게시글 작성

제목

내용

게시글 작성

제목 : 자전거와 함께하는 즐거운 여행!

글쓴이 : admin

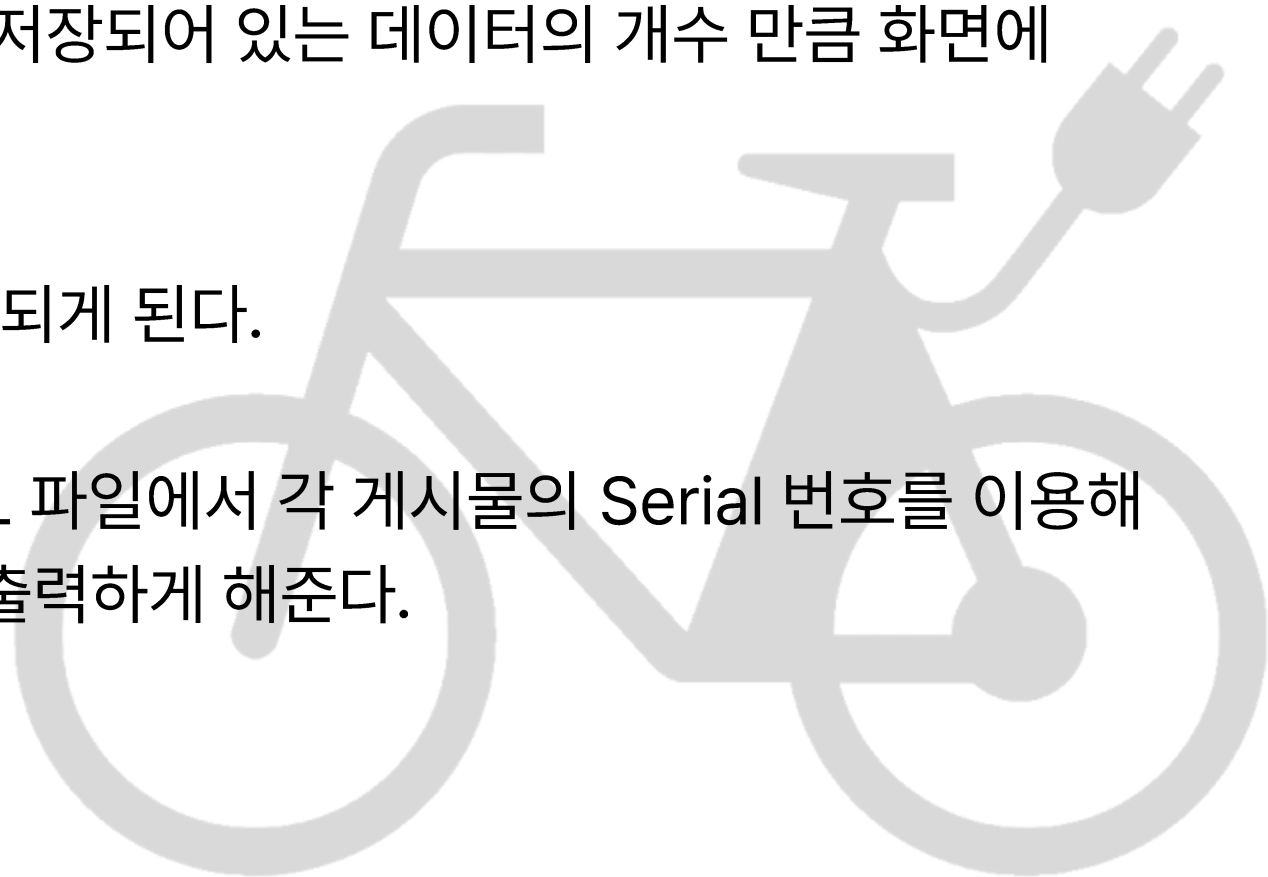
풍경을 감상하며 바람을 맞으며 자전거를 타는 즐거움은 말로 표현하기 어려운 특별한 경험입니다. 자연 속을 휩쓸며 새로운 모험을 찾아나서는 자전거 여행은 마음의 평화를 찾고 새로운 에너지를 얻는 좋은 방법입니다. 친구, 가족 또는 혼자, 자전거와 함께하는 여행은 삶을 즐기는 새로운 방법을 발견하는 기회가 될 것입니다.

작성 시간 : 2024-05-13 00:00:00

게시글로

{ Community[게시판], Write Area[작성란], Reading Page[열람] }

- 1.Community : NOTICE DB에 저장되어 있는 게시판 데이터를 가져와 Thymeleaf를 통해 저장되어 있는 데이터의 개수 만큼 화면에 표시하게 된다.
- 2.Write Area : 작성한 게시물에 대한 데이터를 NOTICE DB에 저장하여 게시판에 바로 추가되게 된다.
- 3.Reading Page : 열람할 수 있는 게시글들은 @ReuquestParam을 이용하여 하나의 HTML 파일에서 각 게시물의 Serial 번호를 이용해 열람하고자 하는 게시물을 클릭했을 때 NOTICE DB에서 해당 데이터를 가져와 웹 페이지에 출력하게 해준다.



05

프로젝트에 대한 개선 방향

{ Spring Security }

처음에는 Spring Security를 사용하여 프로젝트를 시작하고 싶었지만, 기능을 제대로 알지 못하는 상태에서 3일 동안 Spring Security를 사용해보기도 했습니다. 그러나 시간이 지날수록 프로젝트와 Spring Security 간의 조화가 어려워지며, 결국 Security를 제거하고 로그인 폼과 DB를 연동하는 방식으로 개발하게 되었습니다. 이러한 선택으로 인해 아쉬움이 남아 다음 프로젝트에서는 개선할 방향을 모색하고 있습니다.

{ Thymeleaf }

Thymeleaf를 사용하면서 새로운 기능들을 알게 되었고, 프로젝트를 진행하면서 더 많은 기능들을 적용하지 못한 것이 아쉬웠습니다. 처음부터 Thymeleaf를 충분히 활용할 수 있다면 웹 페이지를 더 간결하고 효율적으로 구성할 수 있었을 텐데, 이를 깨달은 것이 프로젝트 중간이라 아쉬웠습니다. 시간을 되돌리고 다시 시작할 수 있다면, Thymeleaf의 다양한 기능을 최대한 활용하여 프로젝트에 더 많은 기능을 더할 수 있을 것입니다. 이러한 경험으로부터 배워서 앞으로의 프로젝트에서는 초기 단계부터 Thymeleaf를 활용하여 더 나은 웹 페이지를 개발하고자 개선 방향으로 잡게 되었습니다.

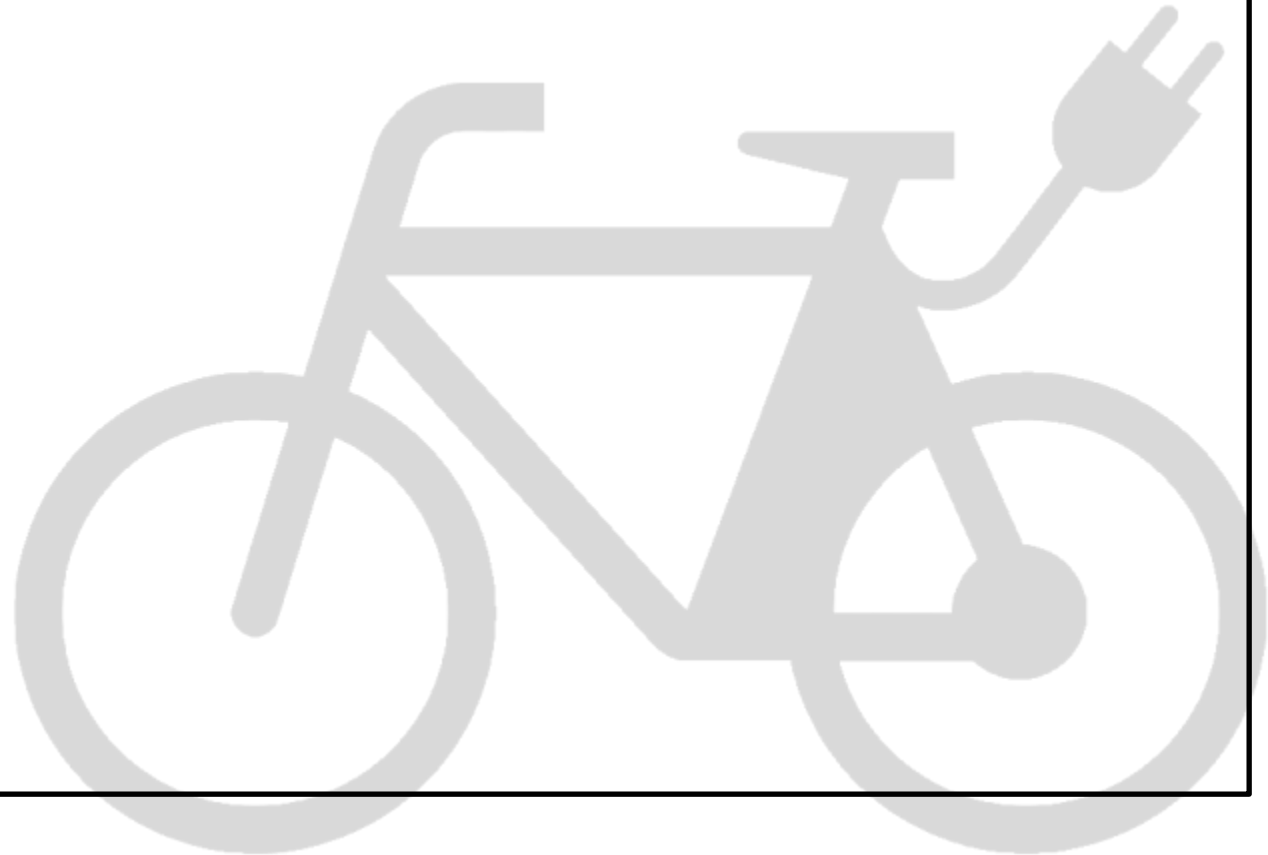
{ HTML & CSS & JS }

프로젝트를 통해 프론트엔드 기술을 다루게 되었지만, 처음에는 너무 복잡해서 포기하고 싶은 생각도 들었습니다. 원하는 기능을 구현하기 위해 어디에 어떻게 적용해야 할지 이해하는 것이 어려웠습니다. 이로 인해 웹 페이지의 퀄리티가 낮아지는 것을 보며 매우 아쉬움을 느꼈습니다. 이러한 경험을 통해 'Simple is best'라는 원칙이 중요하다는 것을 깨달았습니다. 다음 프로젝트에서는 기능을 간결하고 명확하게 구현하여 사용자 경험을 향상시키고자 합니다.

06 마무리하며...

{ Impressions }

처음에는 Spring Security를 사용하여 프로젝트를 시작하고 싶었지만, 기능을 제대로 알지 못하는 상태에서 3일 동안 Spring Security를 사용해보기도 했습니다. 그러나 시간이 지날수록 프로젝트와 Spring Security 간의 조화가 어려워지며, 결국 Security를 제거하고 로그인 폼과 DB를 연동하는 방식으로 개발하게 되었습니다. 이러한 선택으로 인해 아쉬움이 남아 다음 프로젝트에서는 개선할 방향을 모색하고 있습니다.



THANK YOU



이상으로 글로벌
아카데미에서 제작한
프로젝트를
마칩니다.
감사합니다.

- ✿ 작성자 한석희
- ✿ 기획자 한석희
- ✿ 개발자 한석희