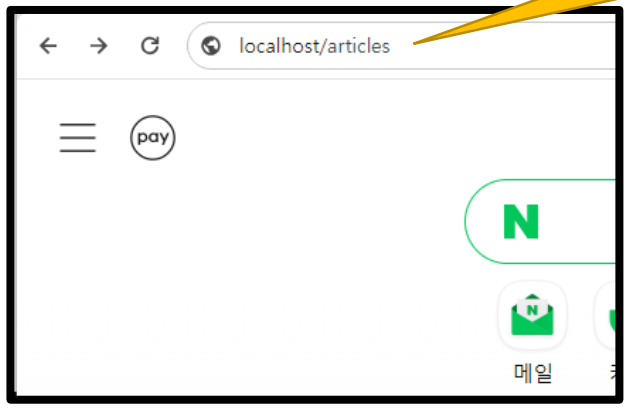
R : read : 전체조회

클라이언트(브라우저)

브라우저에 입력하는것은
HTTP GET 방식입니다.

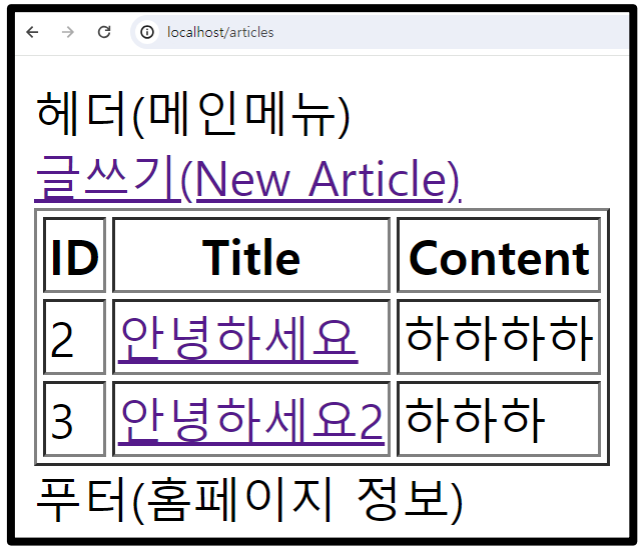


HTTP GET
localhost/articles
요청

서버(스프링)

```
@GetMapping("/articles")
public String index( Model model ){
    // 1. p.175 DAO에게 요청해서 데이터 가져온다.
    List<ArticleForm> result = articleDao.index();
    // 2. p.175 뷰템플릿(머스테치)에게 전달할 값을 model 담아준다.
    model.addAttribute( attributeName: "articleList", result);
    // 3. p.175 뷰 페이지 설정
    return "articles/index";
}
```

머스테치
model.addAttribute("articleList", result);



HTTP GET
localhost/articles
응답

[View (HTML) 렌더링]
머스테치를 HTML 만드는과정(자동)

Index.mustache

C : create

클라이언트(브라우저)

서버(스프링)

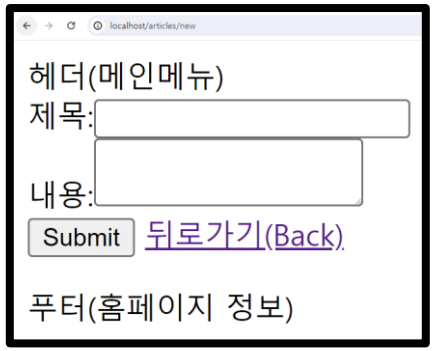
글쓰기(New Article)

<a> 클릭은 HTTP GET 방식입니다.

HTTP GET
localhost/articles/new
요청

```
@GetMapping("/articles/new") // HT
public String newArticleForm(){
    return "articles/new";    // .
} // m end
```

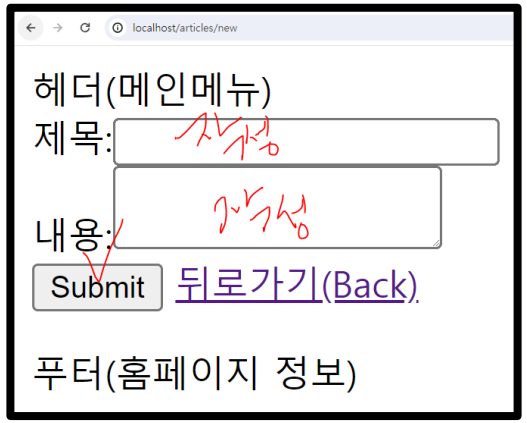
머스테치



HTTP GET
localhost/articles/new
응답

[View (HTML) 렌더링]
머스테치를 HTML 만드는과정(자동)

new.mustache



<form action="/articles/create" method="post">

HTTP POST
localhost/articles/create
요청

```
@PostMapping("/articles/create")    // HTTP 요청 경로 : POST
public String createArticle( ArticleForm form ){

    // DAO에게 요청하고 응답 받기.
    ArticleForm saved = articleDao.createArticle( form );
    return "redirect:/articles/"+saved.getId() ;

} // m end
```

리다이렉트

```
제목:<input name="title" />    <br/>
내용:<textarea name="content"></textarea>  <br/>
```

```
@AllArgsConstructor // 컴파일
@NoArgsConstructor  // 컴파일
@ToString           // 컴파일
@Getter@Setter      // 컴파일
public class ArticleForm {
    // 1. 필드
    private long id;
    private String title; //
    private String content;
}
```

R : Read 개별조회

클라이언트(브라우저)

서버(스프링)

리다이렉트

```
@GetMapping("/articles/{id}")   // 클라이언트 요청 예시 :   /articles/
public String show(@PathVariable Long id , Model model ){
    ArticleForm form = articleDao.show( id );
    model.addAttribute( attributeName: "article" , form );
    model.addAttribute( attributeName: "name" , attributeValue: "유재석" );
    return "articles/show";
}
```

머스테치
model.addAttribute("article", form);

헤더(메인메뉴)
목록보기(Go to Article List)
수정(edit)
삭제(Delete)

ID	Title	Content
4	ddsD	ASDASD

푸터(홈페이지 정보)

HTTP GET
localhost/articles/4
응답

[View (HTML) 렌더링]
머스테치를 HTML 만드는과정(자동)

show.mustache

U : update

클라이언트(브라우저)

서버(스프링)

localhost/articles/4

헤더(메인메뉴)
목록보기(Go to Article List)
수정(edit)

```
<a href="/articles/{{ article.id }}/edit"> 수정(edit)</a> <br/>
```

HTTP GET
localhost/articles/4/edit
요청

```
@GetMapping("/articles/{id}/edit") // GetMapping이유: <a> 이
public String edit( @PathVariable long id , Model model ){
    System.out.println("id = " + id);
    // 1. DAO에게 요청하고 응답 받는다.
    ArticleForm form = articleDao.findById( id );
    // 2. 응답결과를 뷰 템플릿에게 보낼 준비 model
    model.addAttribute( attributeName: "article" , form );
    // 3. 뷰 페이지 설정
    return "articles/edit";
}
```

머스테치
model.addAttribute("article", form);

[View (HTML) 렌더링]
머스테치를 HTML 만드는과정(자동)

edit.mustache

HTTP GET
localhost/articles/4/edit
응답

localhost/articles/4/edit

헤더(메인메뉴)
제목: ddsD
내용: ASDASD
Submit 뒤로가기(Back)

푸터(홈페이지 정보)

```
<form action="/article/update" method="post">
```

HTTP POST
localhost/articles/update
요청

localhost/articles/4/edit

헤더(메인메뉴)
제목: ddsD수정
내용: ASDASD수정
Submit 뒤로가기(Back)

푸터(홈페이지 정보)

```
@PostMapping("/article/update") // @PatchMapping @Put
public String update( ArticleForm form ){
    // * form 입력 데이터를 Dto 매개변수로 받을때
        // 1. form 입력상자의 name 과 Dto의 필드명 동일
        // 2. Dto의 필드 값을 저장할 생상자 필요
    System.out.println("form = " + form);
    // 2.DAO 에게 요청하고 응답받기
    ArticleForm updated = articleDao.update( form );
    // 3. 수정 처리된 상세페이지로 이동
    return "redirect:/articles/"+updated.getId();
}
```

```
<input name="id" type="hidden" value="{{id}}"/>
제목:<input name="title" value="{{ title }}" />  <br/>
내용:<textarea name="content" >{{ content }}</textarea> <br/>
```

```
@AllArgsConstructor // 컴파일
@NoArgsConstructor  // 컴파일
@ToString           // 컴파일
@Getter@Setter      // 컴파일
public class ArticleForm {
    // 1. 필드
    private long id;
    private String title; //
    private String content;//
}
```

리다이렉트

R : Read 개별조회

클라이언트(브라우저)

서버(스프링)

리다이렉트

```
@GetMapping("/articles/{id}")   // 클라이언트 요청 예시 :   /articles/
public String show(@PathVariable Long id , Model model ){
    ArticleForm form = articleDao.show( id );
    model.addAttribute( attributeName: "article" , form );
    model.addAttribute( attributeName: "name" , attributeValue: "유재석" );
    return "articles/show";
}
```

머스테치
model.addAttribute("article", form);

[View (HTML) 렌더링]
머스테치를 HTML 만드는과정(자동)

show.mustache

HTTP GET
localhost/articles/4
응답

localhost/articles/4

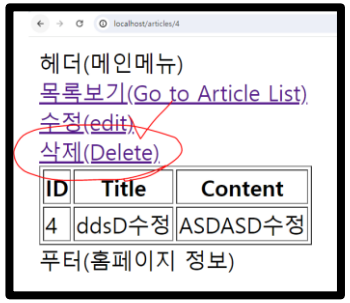
헤더(메인메뉴)
목록보기(Go to Article List)
수정(edit)
삭제(Delete)

ID	Title	Content
4	ddsD	ASDASD

푸터(홈페이지 정보)

D : delete

클라이언트(브라우저)




```
<a href="/articles/{{ article.id }}/delete">삭제(Delete)</a> <br/>
```

HTTP GET
localhost/articles/4/delete
요청

서버(스프링)

```
@GetMapping("/articles/{id}/delete")
public String delete(@PathVariable long id ){
    System.out.println("id = " + id);
    // 1. 삭제할 대상
    // 2. Dao 삭제 요청하고 응답받기
    boolean result = articleDao.delete( id );
    // 3. 결과 페이지로 리다이렉트 하기.
    return "redirect:/articles";
}
```

리다이렉트

```
@GetMapping("/articles")
public String index( Model model ){
    // 1. p.175 DAO에게 요청해서 데이터 가져온다.
    List<ArticleForm> result = articleDao.index();
    // 2. p.175 뷰템플릿(머스테치)에게 전달할 값을 model 담아준다.
    model.addAttribute( attributeName: "articleList", result);
    // 3. p.175 뷰 페이지 설정
    return "articles/index";
}
```

머스테치
model.addAttribute("articleList", result);

[View (HTML) 렌더링]
머스테치를 HTML 만드는과정(자동)

index.mustache

HTTP GET
localhost/articles

HTTP

- URL : HTTP 규약의 통신 식별 경로
- method : HTTP 규약의 통신 방법
  1. GET 2. POST 3. PUT 4. DELETE
- HTTP 이용한 자원 제공방법

1. path : URL 사이의 문자형식 데이터 넣어서 보내는 방법
   1. 요청
      localhost/articles/{매개변수1}/{매개변수2}
      localhost/articles/4/edit
   2. 스프링 받을때.
      매핑함수(@PathVariable 타입 매개변수명){ }

2. contentType : HTTP 제공하는 내용물(payload)을 body 포함해서 보내는방법
   1. application/x-www-form-urlencode < HTML form 기본형태>
      주로 폼 전송시 사용
      1. 요청

```
<form action="/articles/create" method="post">
    제목:<input name="title" />   <br/>
    내용:<textarea name="content"></textarea> <br/>
    <button type="submit">Submit</button>
     <a href="/articles">뒤로가기(Back)</a>
</form>
```

      2. 스프링 받을때

```
매핑함수( 객체타입 매개변수명 ){ }
```

   2. *application/json < JQUERY AJAX 이용한 형태 >

요청HTTP					서버HTTP Request		서버HTTP Response	
기능	URL	params 설명	method	contentType	mapping	params	응답	contentType
전체페이지 요청	/articles		GET	text/html	@GetMapping		머스테치 파일	text/html;
쓰기페이지 요청	/articles/new		GET	text/html	@GetMapping		머스테치 파일	text/html;
쓰기처리 요청	/articles/create	{ title="제목", content="내용" }	POST	application/x-www-form-urlencode	@PostMapping	ArticleForm form	"redirect:/articles/"+saved.getId() ;	
개별페이지 요청	/articles/{id}	id : 게시물 식별키	GET	text/html	@GetMapping	@PathVariable Long id	머스테치 파일	text/html;
수정페이지 요청	/articles/{id}/edit	id : 게시물 식별키	GET	text/html	@GetMapping	@PathVariable long id	머스테치 파일	text/html;
수정처리 요청	/articles/update	{ id="수정할식별키", title="수정할새로운제목", content="수정할새로운내용" }	POST	application/x-www-form-urlencode	@PostMapping	ArticleForm form	"redirect:/articles/"+saved.getId() ;	
삭제처리 요청	/articles/{id}/delete	id : 게시물 식별키	GET	text/html	@GetMapping	@PathVariable long id	"redirect:/articles"	

실습 1

- 회원가입 과 로그인 REST API 구축

회원가입 처리 요청	/member/signup	{ id ="아이디" , pw ="비밀번호" , name="이름" , email="이메일" , phone="전화번호" , img ="프로필사진" }	POST	application/x-www-form-urlencode	@PostMapping	MemberDto dto	boolean : true/false	@ResponseBody : application/json;
로그인 처리 요청	/member/login	{ id="아이디" , pw ="비밀번호" }	POST	application/x-www-form-urlencode	@PostMapping	LoginDto dto	boolean : true/false	@ResponseBody : application/json;
회원가입 페이지 요청	/member/signup		GET	text/html	@GetMapping		머스테치 파일	text/html;
로그인 페이지 요청	/member/login		GET	text/html	@GetMapping		머스테치 파일	text/html;
	* URL 주소 가 같으면 안되지만 method가 다르면 가능* URL 주소 와 method가 동일한 경우는 불가능							

MemberController.java

```java
@Controller
public class MemberController {
    // 1.========== 회원가입 처리 요청 ==========
    @PostMapping("/member/signup") // http://localhost:80/member/signup
    @ResponseBody // 응답 방식 application/json;
    public boolean doPostSignup( MemberDto memberDto ){
        /* params  {  id ="아이디" , pw ="비밀번호" , name="이름" ,  ema
        System.out.println("MemberController.signup");
        System.out.println("memberDto = " + memberDto);
        // --
        boolean result = true;//Dao처리;
        return result; // Dao 요청후 응답 결과를 보내기.
    }
    // 2. ========== 로그인 처리 요청 ==========
    @PostMapping("/member/login") // http://localhost:80/member/login
    @ResponseBody  // 응답 방식 application/json;
    public boolean doPostLogin( LoginDto loginDto ){
        /* params    { id ="아이디" , pw ="비밀번호" }   */
        System.out.println("MemberController.login");
        System.out.println("loginDto = " + loginDto);
        // --
        boolean result = true; // Dao처리
        return result; // Dao 요청후 응답 결과를 보내기
    }
    // 3. ========== 회원가입 페이지 요청 ==========
    @GetMapping("/member/signup")
    public String viewSignup(){
        System.out.println("MemberController.viewSignup");
        return "ezenweb/signup";
    }
    // 4. ========== 로그인 페이지 요청 ==========
    @GetMapping("/member/login")
    public String viewLogin(){
        System.out.println("MemberController.viewLogin");
        return "ezenweb/login";
    }
}
```

LoginDto.java

```java
package ezenweb.model.dto;

import lombok.*;

@AllArgsConstructor
@NoArgsConstructor
@Getter @Setter @ToString
public class LoginDto {
    private int no;          /*회원번호*/
    private String id;
    private String pw;
}
```

MemberDto.java

```java
@AllArgsConstructor
@NoArgsConstructor
@Getter@Setter@ToString
public class MemberDto {
    private int no;          /*회원번호*/
    private String id;
    private String pw;
    private String name;
    private String email;
    private String phone;
    private String img;
}
```

signup.mustache

```html
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, in
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>

    <h3> 회원가입 페이지 </h3>

</body>
</html>
```

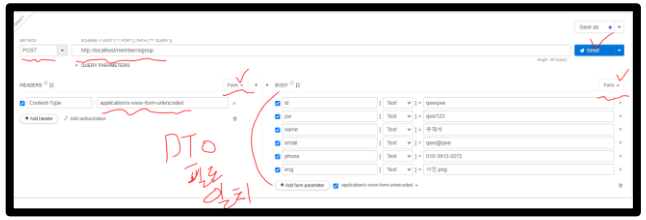
login.mustache

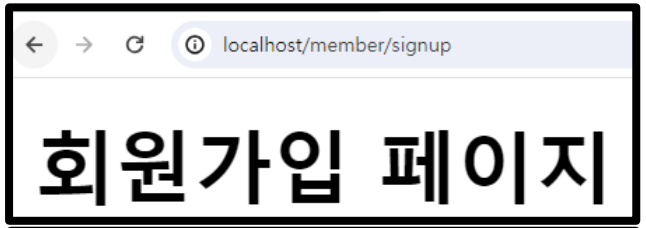
```html
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable
    <meta http-equiv="X-UA-Compatible" content="ie=e
    <title>Document</title>
</head>
<body>

    <h3> 로그인 페이지 </h3>

</body>
</html>
```

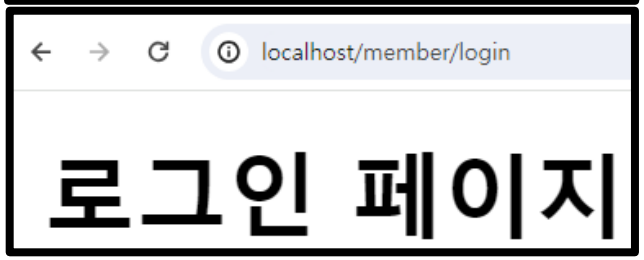
실습 1

- 회원가입 과 로그인 REST API 구축

회원가입 처리 요청	/member/signup	{ id ="아이디" , pw ="비밀번호" , name="이름" , email="이메일" , phone="전화번호" , img ="프로필사진" }	POST	application/x-www-form-urlencode	@PostMapping	MemberDto dto	boolean : true/false	@ResponseBody : application/json;
로그인 처리 요청	/member/login	{ id="아이디" , pw ="비밀번호" }	POST	application/x-www-form-urlencode	@PostMapping	LoginDto dto	boolean : true/false	@ResponseBody : application/json;
회원가입 페이지 요청	/member/signup		GET	text/html	@GetMapping		머스테치 파일	text/html;
로그인 페이지 요청	/member/login		GET	text/html	@GetMapping		머스테치 파일	text/html;
	* URL 주소 가 같으면 안되지만 method가 다르면 가능* URL 주소 와 method가 동일한 경우는 불가능							







MemberController.java

```java
@Controller
public class MemberController {
    // 1.========== 회원가입 처리 요청 ==========
    @PostMapping("/member/signup") // http://localhost:80/member/signup
    @ResponseBody // 응답 방식 application/json;
    public boolean doPostSignup( MemberDto memberDto ){
        /* params  {  id ="아이디" , pw ="비밀번호" , name="이름" ,  ema
        System.out.println("MemberController.signup");
        System.out.println("memberDto = " + memberDto);
        // --
        boolean result = true;//Dao처리;
        return result; // Dao 요청후 응답 결과를 보내기.
    }
    // 2. ========== 로그인 처리 요청 ==========
    @PostMapping("/member/login") // http://localhost:80/member/login
    @ResponseBody  // 응답 방식 application/json;
    public boolean doPostLogin( LoginDto loginDto ){
        /* params    { id ="아이디" , pw ="비밀번호"  }   */
        System.out.println("MemberController.login");
        System.out.println("loginDto = " + loginDto);
        // --
        boolean result = true; // Dao처리
        return result; // Dao 요청후 응답 결과를 보내기
    }
    // 3. ========== 회원가입 페이지 요청 ==========
    @GetMapping("/member/signup")
    public String viewSignup(){
        System.out.println("MemberController.viewSignup");
        return "ezenweb/signup";
    }
    // 4. ========== 로그인 페이지 요청 ==========
    @GetMapping("/member/login")
    public String viewLogin(){
        System.out.println("MemberController.viewLogin");
        return "ezenweb/login";
    }
}
```

LoginDto.java

```java
package ezenweb.model.dto;

import lombok.*;

@AllArgsConstructor
@NoArgsConstructor
@Getter @Setter @ToString
public class LoginDto {
    private int no;          /*회원번호*/
    private String id;
    private String pw;
}
```

MemberDto.java

```java
@AllArgsConstructor
@NoArgsConstructor
@Getter@Setter@ToString
public class MemberDto {
    private int no;          /*회원번호*/
    private String id;
    private String pw;
    private String name;
    private String email;
    private String phone;
    private String img;
}
```

signup.mustache

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, in
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>

    <h3> 회원가입 페이지 </h3>

</body>
</html>
```

login.mustache

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable
    <meta http-equiv="X-UA-Compatible" content="ie=e
    <title>Document</title>
</head>
<body>

    <h3> 로그인 페이지 </h3>

</body>
</html>
```