7 장 완성 코드

templates/articles/show.mustache

```
{{>layouts/header}}

<table class="table">
    <thead>
    <tr>
        <th scope="col">Id</th>
        <th scope="col">Title</th>
        <th scope="col">Content</th>
    </tr>
    </thead>
    <tbody>
    {{#article}}
    <tr>
        <th>{{id}}</th>
        <td>{{title}}</td>
        <td>{{content}}</td>
    </tr>
    {{/article}}
    </tbody>
</table>

<a href="/articles/{{article.id}}/edit" class="btn btn-primary">Edit</a>
<a href="/articles">Go to Article List</a>

{{>layouts/footer}}
```

controller/ArticleController.java

```
package com.example.firstproject.controller;

import com.example.firstproject.dto.ArticleForm;
import com.example.firstproject.entity.Article;
import com.example.firstproject.repository.ArticleRepository;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
```

```java
import org.springframework.web.bind.annotation.PostMapping;

import java.util.List;

@Slf4j
@Controller
public class ArticleController {
    @Autowired
    private ArticleRepository articleRepository;

    @GetMapping("/articles/new")
    public String newArticleForm() {
        return "articles/new";
    }

    @PostMapping("/articles/create")
    public String createArticle(ArticleForm form) {
        log.info(form.toString());
        // System.out.println(form.toString());

        // 1. DTO를 엔티티로 변환
        Article article = form.toEntity();
        log.info(article.toString());
        // System.out.println(article.toString());

        // 2. 리파지터리로 엔티티를 DB에 저장
        Article saved = articleRepository.save(article);
        log.info(saved.toString());
        // System.out.println(saved.toString());

        return "redirect:/articles/" + saved.getId();
    }

    @GetMapping("/articles/{id}") // 데이터 조회 요청 접수
    public String show(@PathVariable Long id, Model model) { // 매개변수로 id 받아오기
        log.info("id = " + id); // id를 잘 받았는지 확인하는 로그 찍기

        // 1. id를 조회하여 데이터 가져오기
        Article articleEntity = articleRepository.findById(id).orElse(null);

        // 2. 모델에 데이터 등록하기
        model.addAttribute("article", articleEntity);

        // 3. 뷰 페이지 반환하기
        return "articles/show";
    }

    @GetMapping("/articles")
    public String index(Model model) {
        // 1. 모든 데이터 가져오기
```

```
        List<Article> articleEntityList = articleRepository.findAll();

        // 2. 모델에 데이터 등록하기
        model.addAttribute("articleList", articleEntityList);

        // 3. 뷰 페이지 설정하기
        return "articles/index";
    }

    @GetMapping("/articles/{id}/edit")
    public String edit(@PathVariable Long id, Model model) {
        // 수정할 데이터 가져오기
        Article articleEntity = articleRepository.findById(id).orElse(null);

        // 모델에 데이터 등록하기
        model.addAttribute("article", articleEntity);

        // 뷰 페이지 설정하기
        return "articles/edit";
    }

    @PostMapping("/articles/update")
    public String update(ArticleForm form) {
        log.info(form.toString());

        // 1. DTO를 엔티티로 변환하기
        Article articleEntity = form.toEntity();
        log.info(articleEntity.toString());

        // 2. 엔티티를 DB로 저장하기
        // 2-1. DB에서 기존 데이터 가져오기
        Article                          target                          =
articleRepository.findById(articleEntity.getId()).orElse(null);

        // 2-2. 기존 데이터 값을 갱신하기
        if (target != null) {
            articleRepository.save(articleEntity);  // 엔티티를 DB에 저장(갱
신)
        }

        // 3. 수정 결과 페이지로 리다이렉트 하기
        return "redirect:/articles/" + articleEntity.getId();
    }
}
```

templates/articles/edit.mustache

```
{{>layouts/header}}

{{#article}}
<form class="container" action="/articles/update" method="post">
    <input name="id" type="hidden" value="{{id}}">
    <div class="mb-3">
        <label class="form-label">제목</label>
        <input       type="text"       class="form-control"       name="title"
value="{{title}}">
    </div>
    <div class="mb-3">
        <label class="form-label">내용</label>
        <textarea                class="form-control"                rows="3"
name="content">{{content}}</textarea>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
    <a href="/articles/{{id}}">Back</a>
</form>
{{/article}}

{{>layouts/footer}}
```

templates/articles/new.mustache

```
{{>layouts/header}}

<form class="container" action="/articles/create" method="post">
    <div class="mb-3">
        <label class="form-label">제목</label>
        <input type="text" class="form-control" name="title">
    </div>
    <div class="mb-3">
        <label class="form-label">내용</label>
        <textarea class="form-control" rows="3" name="content"></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
    <a href="/articles">Back</a>
</form>

{{>layouts/footer}}
```

resources/data.sql

```
INSERT INTO article(id, title, content) VALUES(1, '가가가가', '1111');
INSERT INTO article(id, title, content) VALUES(2, '나나나나', '2222');
INSERT INTO article(id, title, content) VALUES(3, '다다다다', '3333');
```

dto/ArticleForm.java

```
package com.example.firstproject.dto;

import com.example.firstproject.entity.Article;
import lombok.AllArgsConstructor;
import lombok.ToString;

@AllArgsConstructor
@ToString
public class ArticleForm {
    private Long id;
    private String title; // 제목을 받을 필드
    private String content; // 내용을 받을 필드

    public Article toEntity() {
        return new Article(id, title, content);
    }
}
```

셀프체크 정답

controller/MemberController.java

```
package com.example.firstproject.controller;

import com.example.firstproject.entity.Member;
import com.example.firstproject.dto.MemberForm;
import com.example.firstproject.repository.MemberRepository;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

@Slf4j
@Controller
public class MemberController {

    @Autowired
    MemberRepository memberRepository;

    @GetMapping("/signup")
    public String signUpPage() {
        return "members/new";
    }

    @PostMapping("/join")
    public String join(MemberForm memberForm) {
        log.info(memberForm.toString());
        Member member = memberForm.toEntity();
        log.info(member.toString());
        Member saved = memberRepository.save(member);
        log.info(saved.toString());
        return "redirect:/members/" + saved.getId();
    }

    @GetMapping("/members/{id}")
    public String show(@PathVariable Long id, Model model) {
        Member member = memberRepository.findById(id).orElse(null);
        model.addAttribute("member", member);
        return "members/show";
    }

    @GetMapping("/members")
    public String index(Model model) {
```

```
        Iterable<Member> members = memberRepository.findAll();
        model.addAttribute("members", members);
        return "members/index";
    }

    @GetMapping("/members/{id}/edit")
    public String edit(@PathVariable Long id, Model model) {
        Member memberEntity = memberRepository.findById(id).orElse(null);
        model.addAttribute("member", memberEntity);
        return "members/edit";
    }

    @PostMapping("/members/update")
    public String update(MemberForm form) {
        log.info(form.toString());
        Member memberEntity = form.toEntity();
        Member target =
memberRepository.findById(memberEntity.getId()).orElse(null);
        if (target != null) {
            memberRepository.save(memberEntity);
        }
        return "redirect:/members/" + memberEntity.getId();
    }
}
```

dto/MemberForm.java

```
package com.example.firstproject.dto;

import com.example.firstproject.entity.Member;
import lombok.AllArgsConstructor;
import lombok.ToString;

@AllArgsConstructor
@ToString
public class MemberForm {
    private Long id;
    private String email;
    private String password;

    public Member toEntity() {
        return new Member(id, email, password);
    }
}
```

templates/members/show.mustache

```
{{>layouts/header}}
<table class="table">
    <thead>
    <tr>
        <th scope="col">Id</th>
        <th scope="col">Email</th>
        <th scope="col">Password</th>
    </tr>
    </thead>
    <tbody>
    {{#member}}
        <tr>
            <th>{{id}}</th>
            <td>{{email}}</td>
            <td>{{password}}</td>
        </tr>
    {{/member}}
    </tbody>
</table>

<a href="/members/{{member.id}}/edit" class="btn btn-primary">수정하기</a>
<a href="/members">Go to member list</a>

{{>layouts/footer}}
```

templates/members/edit.mustache

```
{{>layouts/header}}

{{#member}}
    <form class="container" action="/members/update" method="post">
        <input type="hidden" name="id" value="{{id}}">
        <div class="mb-3">
            <label class="form-label">이메일</label>
            <input type="email" class="form-control" name="email"
                   value="{{email}}">
        </div>
        <div class="mb-3">
            <label class="form-label">비밀번호</label>
            <input type="password" class="form-control" name="password"
                   value="{{password}}">
        </div>
        <button type="submit" class="btn btn-primary">수정완료</button>
        <a href="/members">Back</a>
    </form>
{{/member}}
```

```
{{>layouts/footer}}
```