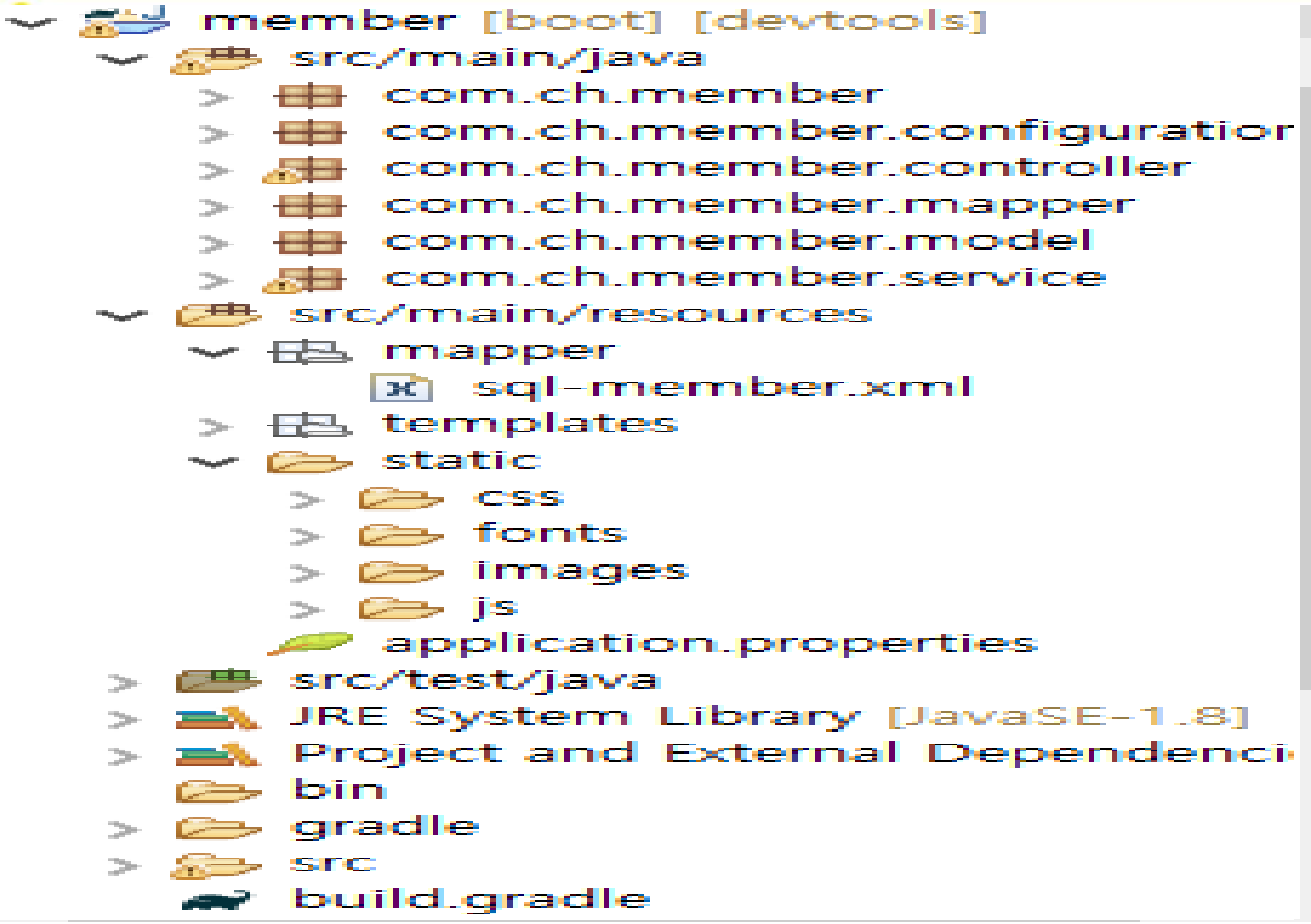
Spring-boot

강병준

회원관리 (Oracle)

```
create table member4 (
        id varchar2(20) primary key,
        email varchar2(30),
        password varchar2(30),
        name varchar2(30),
        fileName varchar2(50) default 'a.jpg',
        del char(1) default 'n',
        regdate date
);
```

구조


member [boot] [devtools]
src/main/java
com.ch.member
com.ch.member.configuratior
com.ch.member.controller
com.ch.member.mapper
com.ch.member.model
com.ch.member.service
src/main/resources
mapper
sql-member.xml
templates
static
css
fonts
images
js
application.properties
src/test/java
JRE System Library [JavaSE-1.8]
Project and External Dependenci
bin
gradle
src
build.gradle

application.properties

```
spring.datasource.hikari.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.hikari.jdbc-url=jdbc:oracle:thin:@127.0.0.1:1521:xe
spring.datasource.hikari.username=scott
spring.datasource.hikari.password=tiger

mybatis.configuration.map-underscore-to-camel-case=true
# jsp, js, css변경할 때 바로적용
spring.devtools.livereload.enabled=true
spring.thymeleaf.cache=false
spring.devtools.restart.enabled=false
```

build.gradle

```
dependencies {
	implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
	implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
	implementation 'org.springframework.boot:spring-boot-starter-web'
	implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.1.3'
	implementation 'commons-io:commons-io:2.6'
	implementation 'commons-fileupload:commons-fileupload:1.4'
	compileOnly 'org.projectlombok:lombok'
	developmentOnly 'org.springframework.boot:spring-boot-devtools'
	runtimeOnly 'com.oracle.database.jdbc:ojdbc8'
	annotationProcessor 'org.springframework.boot:spring-boot-configuration-
processor'
	annotationProcessor 'org.projectlombok:lombok'
	testImplementation('org.springframework.boot:spring-boot-starter-test') {
		exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
	}
}

test {
	useJUnitPlatform()
}
```

Member

```
package com.ch.member.model;
import java.sql.Date;
import org.springframework.web.multipart.MultipartFile;
import lombok.Data;
@Data
public class Member {
		private String id;
		private String email;
		private String password;
		private String name;
		private String fileName;
		private String del;
		private Date regdate;
		// upload 사진용
		private MultipartFile file;
}
```

MemberDao

```
package com.ch.member.mapper;
import org.apache.ibatis.annotations.Mapper;
import com.ch.member.model.Member;
@Mapper
public interface MemberDao {
        Member select(String id);
        int insert(Member member);
        int update(Member member);
        int delete(String id);
}
```

MemberServiceImpl

```
package com.ch.member.service;
@Service
public class MemberServiceImpl implements MemberService{
	@Autowired
	private MemberDao md;
	public Member select(String id) {
		return md.select(id);
	}
	public int update(Member member) {
		return md.update(member);
	}
	public int delete(String id) {
		return md.delete(id);
	}
	public int insert(Member member) {
		return md.insert(member);
	}
}
```

application.properties

```
spring.datasource.hikari.driver-class-name=oracle.jdbc.driver.OracleDriver

spring.datasource.hikari.jdbc-url=jdbc:oracle:thin:@127.0.0.1:1521:xe

spring.datasource.hikari.username=scott

spring.datasource.hikari.password=tiger


mybatis.configuration.map-underscore-to-camel-case=true

spring.devtools.livereload.enabled=true

spring.thymeleaf.cache=false

spring.devtools.restart.enabled=false
```

sql-member.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.ch.member.mapper.MemberDao">
<select id="select" parameterType="string"
        resultType="com.ch.member.model.Member">
        select * from member4 where id=#{id}
</select>
<insert id="insert" parameterType="com.ch.member.model.Member">
        insert into member4 values (#{id},#{email},#{password},
                #{name},#{fileName},'n',sysdate)
</insert>
<update id="update" parameterType="com.ch.member.model.Member">
        update member4 set email=#{email}, password=#{password},
                name=#{name}
                <!-- 파일을 변경하지 않으면 먼저 파일 사용 -->
                <if test="fileName!=null“>,fileName=#{fileName}</if>
                where id=#{id}
</update>
<update id="delete" parameterType="string">
        update member4 set del='y' where id=#{id}
</update>
</mapper>
```

WebMvcConfiguration

```
package com.ch.member.configuration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.multipart.commons.CommonsMultipartResolver;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import com.ch.member.service.SessionChk;
@Configuration public class WebMvcConfiguration implements
WebMvcConfigurer{
    @Bean
   public CommonsMultipartResolver multipartResolver(){
            CommonsMultipartResolver commonsMultipartResolver =
                    new CommonsMultipartResolver();
            commonsMultipartResolver.setDefaultEncoding("UTF-8");
            commonsMultipartResolver.setMaxUploadSizePerFile(
                    10 * 1024 * 1024); //10 * 1024 * 1024 (5mb)
            return commonsMultipartResolver;
   }
```

WebMvcConfiguration

```
    public void addInterceptors(InterceptorRegistry ir) {
        List<String> URL_PATTERNS =
        Arrays.asList("/**/main.do","/**/updateForm.do",
                "/**/update.do","/**/delete.do","/**/view.do");
        ir.addInterceptor(new SessionChk())
                .addPathPatterns(URL_PATTERNS);
//              .addPathPatterns("/**/*.do")
//              .excludePathPatterns(URL_PATTERNS");
    }
}
```

MemberController

```
package com.ch.member.controller;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.List;
import javax.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartHttpServletRequest;
import com.ch.member.model.Member;
import com.ch.member.service.MemberService;
@Controller
public class MemberController {
		@Autowired
		private MemberService ms;
		@RequestMapping("joinForm.do")
		public String joinForm2() {
				return "joinForm";
		}
```

MemberController

```
@RequestMapping(value="idChk.do",produces="text/html;charset=utf-8")
@ResponseBody  // jsp프로그램을 이용하지 않고 바로 본문을 전달
public String idChk(String id) {
		String msg = "";
		Member member = ms.select(id);
		if (member == null) msg="사용가능한 ID입니다";
		else msg="이미 있으니 다른 ID를 사용하시오";
		return msg;
}
@RequestMapping("loginForm.do")
public String loginForm() {	return "loginForm“;  }
@RequestMapping("login.do")
public String login(Member member,Model model,HttpSession session) {
		int result = 0;
		Member mem = ms.select(member.getId());
		if (mem == null || mem.getDel().equals("y")) result = -1;  // 없는 ID
		else if (mem.getPassword().equals(member.getPassword())) {
				result = 1; // 성공
				session.setAttribute("id", member.getId());
		}
		model.addAttribute("result", result);
		return "login";
}
```

MemberController

```
@RequestMapping("main.do")
public String main(Model model, HttpSession session) {
        String id = (String)session.getAttribute("id");
        Member member = null;
        if (id != null && !id.equals(""))
                member = ms.select(id);
        model.addAttribute("member", member);
        return "main";
}
@RequestMapping("view.do")
public String view2(Model model, HttpSession session) {
        String id = (String)session.getAttribute("id");
        Member member = ms.select(id);
        model.addAttribute("member", member);
        return "view";
}
@RequestMapping("logout.do")
public String logout(HttpSession session) {
        session.invalidate();
        return "logout";
}
```

MemberController

```
@RequestMapping("updateForm.do")
	public String updateForm(Model model, HttpSession session) {
		String id = (String)session.getAttribute("id");
		Member member = ms.select(id);
		model.addAttribute("member", member);
		return "updateForm";
	}
	@RequestMapping("update.do")
	public String update(Member member,Model model,HttpSession session) throws
IOException {
		String fileName = member.getFile().getOriginalFilename();
		// 수정에서 파일을 선택하지 않으면 값이 공란("")이 넘어옴
		if (fileName!=null && !fileName.equals("")) {
			FileOutputStream fos =
				new FileOutputStream(new
File("src/main/resources/static/images/"+fileName));
			fos.write(member.getFile().getBytes());
			fos.close();
			member.setFileName(fileName);
		}
		int result = ms.update(member);
		model.addAttribute("result", result);
		return "update";
	}
```

MemberController

```
@RequestMapping("delete.do")
public String delete(Model model, HttpSession session) {
        String id = (String)session.getAttribute("id");
        int result  = ms.delete(id);
        if (result > 0 ) session.invalidate();
        model.addAttribute("result", result);
        return "delete";
}
```

MemberController

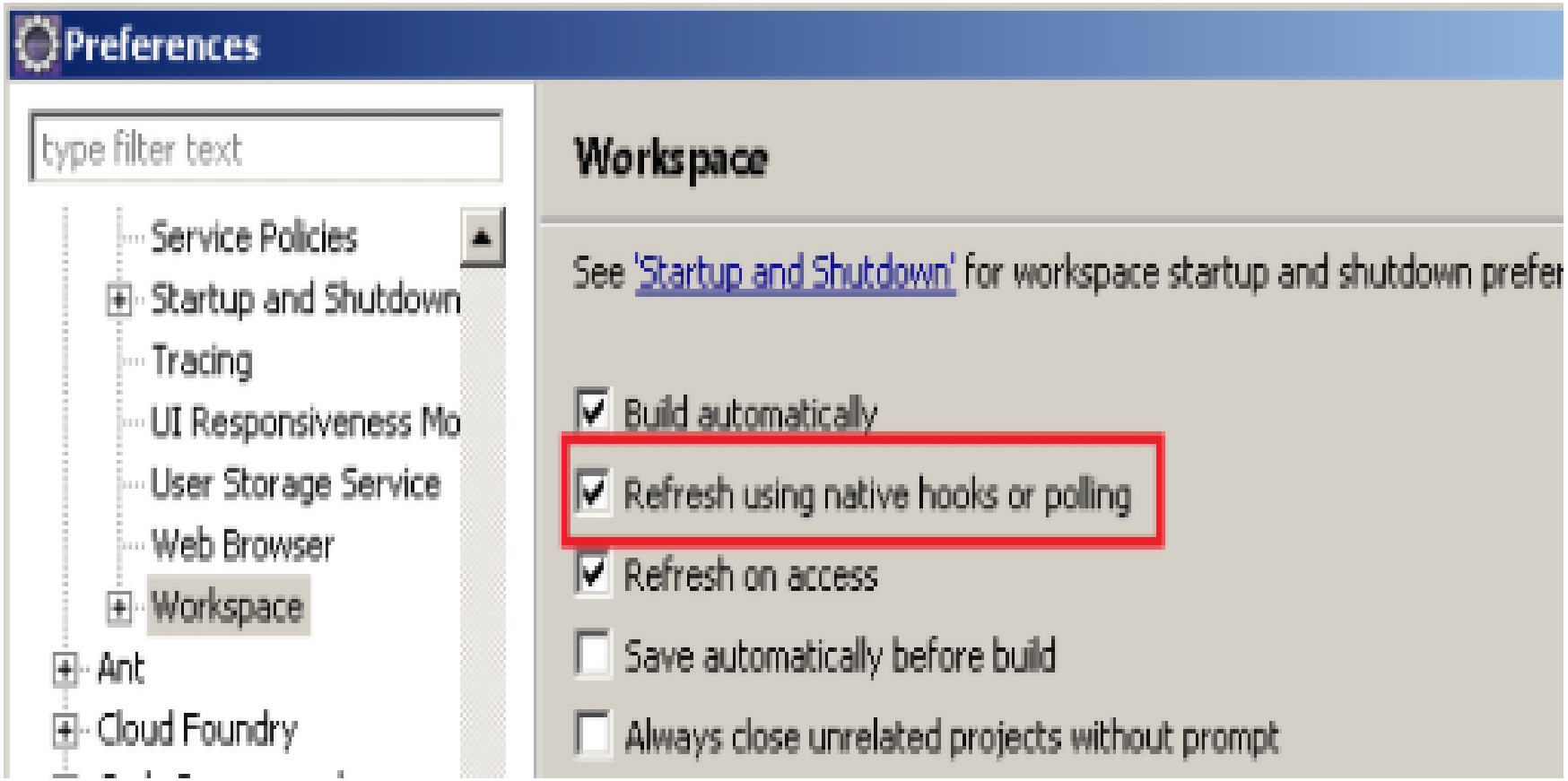
```
    @RequestMapping("join.do")
    public String join(Member member,
                    Model model,HttpSession session) throws IOException {
        int result = 0;
        Member mem = ms.select(member.getId());
        if (mem == null) {
                // 화면에서 member로 파일명은 입력되지 않음
                String fileName=member.getFile().getOriginalFilename);
                member.setFileName(fileName);
                FileOutputStream fos =
                        new FileOutputStream(new
                        File("src/main/resources/static/images/"+fileName));
                fos.write(member.getFile().getBytes());
                fos.close();
                result = ms.insert(member);
        } else result = -1;
        model.addAttribute("result", result);
        return "join";
    }
}
```

SessionChk

```
package com.ch.member.service;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.springframework.web.servlet.HandlerInterceptor;
public class SessionChk implements HandlerInterceptor {
        public boolean preHandle(HttpServletRequest request,
HttpServletResponse response, Object handler)
                        throws Exception {
                HttpSession session = request.getSession();
                if (session == null ||
                                session.getAttribute("id") == null ||
                                session.getAttribute("id").equals("")) {
                        response.sendRedirect("loginForm.do");
                        return false;
                } return true;
        }
}
```

Upload한 그림 즉시 반영

"Refresh using native hooks or polling"



joinForm.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<title>Insert title here</title>
<script type="text/javascript">
		function idChk() {
			if (!frm.id.value) { alert("아이디 입력하고 중복체크하시오");
				frm.id.focus(); return false;				}
			$.post('idChk.do','id='+frm.id.value, function(data) {
				// alert(data);
				$('#disp').html(data);				});
		}
		function chk() {
			if (frm.password.value != frm.password2.value) {
				alert("암호와 암호 확인이 다릅니다");
				frm.password.focus(); frm.password.value="";
				return false;
			}
		}
</script></head><body>
```

joinForm.html

```
<div class="container" align="center">
<form action="join.do" method="post" name="frm"
        onsubmit="return chk()" enctype="multipart/form-data">
<table class="table table-bordered">
        <caption class="text-primary">회원 가입</caption>
        <tr><td>아이디 <span class="glyphicon glyphicon-user"></span></td>
                <td><input type="text" name="id" required="required"
                autofocus="autofocus"><input type="button" onclick="idChk()"
                class="btn btn-info btn-sm" value="중복체크"><span id="disp"
class="err"></span></td></tr>
        <tr><td>암호 <span class="glyphicon glyphicon-lock"></span></td>
<td><input type="password" name="password“      required="required"></td></tr>
        <tr><td>암호 확인<span class="glyphicon glyphicon-lock"></span></td>
<td><input type="password" name="password2“    required="required"></td></tr>
        <tr><td>이름</td><td><input type="text" name="name"
                        required="required"></td></tr>
        <tr><td>이메일 <span class="glyphicon glyphicon-envelope"></span></td>
                <td><input type="email" name="email" required="required"></td></tr>
        <tr><td>사진 <span class="glyphicon glyphicon-picture"></span></td>
                <td><input type="file" name="file" required="required"></td></tr>
        <tr><th colspan="2"><input type="submit"></th></tr>
</table></form>
<a href="loginForm.do" class="btn btn-default">로그인</a>
</div></body></html>
```

Join.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<title>Insert title here</title></head><body>
<span th:if="${result} > 0">
		<script type="text/javascript">
				alert("입력 완료 □□");
				location.href="loginForm.do“;  </script>
</span>
<span th:if="${result} == 0">
		<script type="text/javascript">
				alert("입력 실패 □□");
				history.go(-1);		</script>
</span>
<span th:if="${result} == -1">
		<script type="text/javascript">
				alert("아이디 중복이라는 데, 왜 입력해");
				history.go(-1);
		</script>
</span></body></html>
```

loginForm.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<title>Insert title here</title></head><body>
<div class="container">
<form action="login.do">
<table class="table table-hover">
		<caption>로그인</caption>
		<tr><td>아이디 <span class="glyphicon glyphicon-user"></span></td>
			<td><input type="text" name="id"
			required="required" autofocus="autofocus"></td></tr>
		<tr><td>암호 <span class="glyphicon glyphicon-lock"></span></td>
			<td><input type="password" name="password"
			required="required"></td></tr>
		<tr><td colspan="2"><input type="submit"></td></tr>
</table>
		<a href="joinForm.do" class="btn btn-info">회원가입</a>
</form>
</div></body></html>
```

Login.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<title>Insert title here</title></head><body>
<th:if test="${result > 0 }">
		<script type="text/javascript">
			alert("로그인 성공");
			location.href="main.do“;		</script>
</th:if>
<th:if test="${result == 0 }">
		<script type="text/javascript">
			alert("암호를 몰라 ! 꺼져 !!");
			history.go(-1);		</script>
</th:if>
<th:if test="${result == -1 }">
		<script type="text/javascript">
			alert("넌 누구냐!! 간첩이지");
			history.go(-1);
		</script>
</th:if></body></html>
```

main.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<title>Insert title here</title></head><body>
<div class="container">
		<h2><span th:text="${member.name}"></span>님 환영</h2>
<table class="table table-hover">
		<tr><th><a href="view.do" class="btn btn-info">조회</a></th>
		<tr><th><a href="updateForm.do" class="btn btn-warning">수정</a></th>
		<tr><th><a href="delete.do" class="btn btn-danger">탈퇴</a></th>
		<tr><th><a href="logout.do" class="btn btn-success">로그아웃</a></th>
</table>
</div>
</body>
</html>
```

view.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<title>Insert title here</title></head><body>
<div class="container">
<table class="table table-striped table-bordered">
		<caption>회원정보 상세</caption>
		<tr><td>아이디</td><td th:text="${member.id}"></td></tr>
		<tr><td>이름</td><td th:text="${member.name}"></td></tr>
		<tr><td>이메일</td><td th:text="${member.email}"></td></tr>
		<tr><td>등록일</td><td th:text="${member.regdate}"></td></tr>
		<tr><td colspan="2">사진</td></tr>
		<tr><td colspan="2">
		<tr><td><img  th:src="@{/images/}+${member.fileName}"
					width="200"></td></tr>
</table>
<a href="main.do" class="btn btn-default">메인</a>
</div>
</body></html>
```

updateForm.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<script type="text/javascript">
		function chk() {
			if (frm.password.value != frm.password2.value) {
				alert("암호와 암호 확인이 다릅니다");
				frm.password.focus();
				frm.password.value = "";
				return false;
			}
		}
</script></head><body><div class="container">
<form action="update.do" method="post" enctype="multipart/form-data"
		name="frm" onsubmit="return chk()">
		<input type="hidden" name="id" th:value="${member.id}">
```

updateForm.html

```
<table class="table table-bordered">
        <caption class="text-primary">회원정보 수정</caption>
        <tr><td>아이디 <span class="glyphicon glyphicon-user"></span></td>
            <td th:text="${member.id}"></td></tr>
        <tr><td>암호 <span class="glyphicon glyphicon-lock"></span></td>
            <td><input type="password" name="password"
                    required="required" autofocus="autofocus"></td></tr>
        <tr><td>암호 확인<span class="glyphicon glyphicon-lock"></span></td>
            <td><input type="password" name="password2"
                    required="required"></td></tr>
        <tr><td>이름</td><td><input type="text" name="name"
                    required="required" th:value="${member.name}"></td></tr>
        <tr><td>이메일 <span class="glyphicon glyphicon-envelope"></span></td>
            <td><input type="email" name="email" required="required"
                    th:value="${member.email}"></td></tr>
        <tr><td>사진 <span class="glyphicon glyphicon-picture"></span></td>
            <td><input type="file" name="file"></td></tr>
        <tr><td colspan="2"><img width="200" th:src="@{/images/}+${member.fileName}">
            </td></tr>
        <tr><th colspan="2"><input type="submit"></th></tr>
</table></form>
</div>
</body></html>
```

update.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<title>Insert title here</title></head><body>
<span th:if="${result} > 0">
		<script type="text/javascript">
			alert("수정 성공 □□");
			location.href="view.do";
		</script>
</span>
<span th:if="${result} == 0">
		<script type="text/javascript">
			alert("짜샤! 똑바로 해");
			history.go(-1);
		</script>
</span>
</body></html>
```

delete.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<title>Insert title here</title></head><body>
<span th:if="${result} > 0">
		<script type="text/javascript">
			alert("탈퇴했으니 꺼져");
			location.href="loginForm.do";
		</script>
</span>
<span th:if="${result} == 0">
		<script type="text/javascript">
			alert("탈퇴 실패 !! 평생 노예해야지");
			history.go(-1);
		</script>
</span>
</body></html>
```

logout.html

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<title>Insert title here</title></head><body>
<script type="text/javascript">
alert("로그아웃 되었습니다");
location.href="loginForm.do";
</script>
</body>
</html>
```

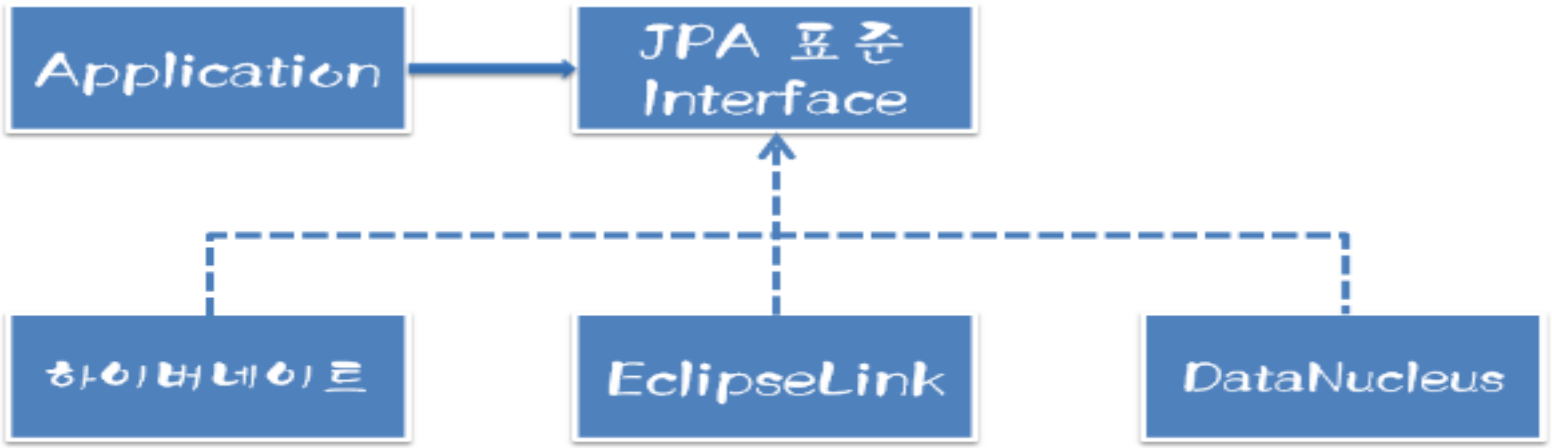
ORM과 JPA

- Object-Relational Mapping (객체와 관계형데이터베이스 매핑, 객체와 DB의 테이블이 매핑을 이루는 것)
- 객체가 테이블이 되도록 매핑 시켜주는 프레임워크 이다.
- 프로그램의 복잡도를 줄이고 자바 객체와 쿼리를 분리할 수 있으며 트랜잭션 처리나 기타 데이터베이스 관련 작업들을 좀 더 편리하게 처리할 수 있는 방법
- SQL Query가 아닌 직관적인 코드(메서드)로서 데이터를 조작할 수 있다.
ex) 기존쿼리 : SELECT * FROM MEMBER; 이를 ORM을 사용하면 Member테이블과 매핑된 객체가 member라고 할 때, member.findAll()이라는 메서드 호출로 데이터 조회가 가능하다.

- Java Persistence API (자바 ORM 기술에 대한 API 표준 명세)
- 한마디로 ORM을 사용하기 위한 인터페이스를 모아둔 것 이라고 볼 수 있다.
- 자바 어플리케이션에서 관계형 데이터베이스를 사용하는 방식을 정의한 인터페이스이다.
- ORM에 대한 자바 API 규격이며 Hibernate, OpenJPA 등이 JPA를 구현한 구현체 이다.
- Hibernate 이외에도 EcipseLink, DataNucleus, OpenJPA, TopLink 등이 있습니다.
※결국 인터페이스이기 때문에 JPA를 사용하기 위해서는 JPA를 구현한 Hibernate, EclipseLink, DataNucleus 같은 ORM 프레임워크를 사용해야 한다.

Hibernate?

- JPA를 사용하기 위해서 JPA를 구현한 ORM 프레임워크중 하나. (자바를 위한 오픈소스 ORM(Object-relational mapping) 프레임워크를 제공한다.)
 - Hibernate는 JPA 명세의 구현체이다. javax.persistence.EntityManager와 같은 JPA의 인터페이스를 직접 구현한 라이브러리이다.
우리가 알고 있는 Hibernate가 JPA의 구현체 인 것이다.



JPA Interface : 인터페이스

JPA장점

1) 생산성이 뛰어나고 유지보수가 용이하다.
(데이터베이스 중심 설계에서 객체 중심 설계로 변경됨에 따른)
- 객체 지향적인 코드로 인해 더 직관적이고 비즈니스 로직에 더 집중할 수 있게 도와준다.
- 객체지향적으로 데이터를 관리할 수 있기 때문에 전체 프로그램 구조를 일관되게 유지할 수 있다.
- SQL을 직접적으로 작성하지 않고 객체를 사용하여 동작하기 때문에 유지보수가 더욱 간결하고, 재사용성도 증가하여 유지보수가 편리해진다.
- DB 컬럼이 추가될 때마다 테이블 수정이나 SQL 수정하는 과정이 많이 줄어들고, 값을 할당하거나, 변수 선언 등의 부수적인 코드 또한 급격히 줄어든다.
- 각각의 객체에 대한 코드를 별도로 작성하여 코드의 가독 성도 올라간다.

2) DBMS에 대한 종속성이 줄어든다.
- DBMS가 변경된다 하더라도 소스, 쿼리, 구현 방법, 자료형 타입 등을 변경할 필요가 없다.
- 즉 프로그래머는 Object에만 집중하면 되고, DBMS를 교체하는 작업에도 비교적 적은 리스크와 시간이 소요된다.
특히 요즘은 탈Oracle을 하여 MariaDB 등의 무료, 오픈소스 기반의 DMBS로 변경하는 기업이 늘고 있는데 이럴 때 개발자들이 신경 쓸 부분이 현저히 줄어든다.

JPA단점

1) 어렵다.
- JPA의 장점을 살려 잘 사용하려면 학습 비용이 높은 편이다.
- 복잡한 쿼리를 사용해야 할 때에 불리하다. 업무 비즈니스가 매우 복잡한 경우 JPA로 처리하기 어렵고, 통계처리와 같은 복잡한 쿼리 자체를 ORM으로 표현하는데 한계가 있다.
(실시간 처리용 쿼리에 더 최적화되어 있다고 봐도 무방할 것이다.)
- 결국 기존 데이터베이스 중심으로 되어 있는 환경에서는 JPA를 사용하기도 어렵고, 힘을 발휘하기 어렵다.
- 잘못 사용할 경우 실제 SQL문을 직접 작성하는 것보다는 성능이 비교적 떨어질 수 있다.
- 대용량 데이터 기반의 환경에서도 튜닝이 어려워 상대적으로 기존 방식보다 성능이 떨어질 수 있다.

결국 업무 환경, 이러한 장단점을 고려하여 Mybatis를 사용할지 JPA를 사용할지 의사결정에 참고하면 좋을 것 같다.

JPA 프로젝트

```
jpa
plugins {
    id 'org.springframework.boot' version '2.6.1'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
    id 'java'
}
group = 'com.ch'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'
configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}
repositories {
    mavenCentral()
}
```

JPA 프로젝트

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.2.0'
    implementation 'org.springframework.boot:spring-boot-starter-jdbc'
    implementation 'org.hibernate.javax.persistence:hibernate-jpa-2.2-
api:1.0.0.Beta2'
    implementation 'org.hibernate:hibernate-entitymanager:5.3.7.Final'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.oracle.database.jdbc:ojdbc8'
    runtimeOnly 'com.h2database:h2'
    annotationProcessor 'org.springframework.boot:spring-boot-configuration-
processor'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}
test {
    useJUnitPlatform()
}
```

applicationContext.properties

```
spring.datasource.hikari.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.hikari.jdbc-url=jdbc:oracle:thin:@127.0.0.1:1521:xe
spring.datasource.hikari.username=scott
spring.datasource.hikari.password=tiger
mybatis.configuration.map-underscore-to-camel-case=true
spring.devtools.livereload.enabled=true
spring.thymeleaf.cache=false
spring.devtools.restart.enabled=false

spring.jpa.database=oracle
spring.jpa.open-in-view=false
```

DatabaseConfiguration

```
package com.ch.jpa.configuration;
import java.util.Properties;
import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.orm.jpa.JpaDialect;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
import org.springframework.orm.jpa.vendor.HibernateJpaDialect;
import org.springframework.orm.jpa.vendor.HibernateJpaSessionFactoryBean;
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
```

DatabaseConfiguration

```
@Configuration
@PropertySource("classpath:/application.properties")
public class DatabaseConfiguration {
    @Autowired
    private ApplicationContext applicationContext;

    @Bean
    @ConfigurationProperties(prefix="spring.datasource.hikari")
    public HikariConfig hikariConfig() {
        return new HikariConfig();
    }
    @Bean
    public DataSource dataSource() {
        DataSource dataSource = new HikariDataSource(hikariConfig());
        return dataSource;
    }
    @Bean
    @ConfigurationProperties(prefix="spring.jpa")
    public Properties hibernateConfig(){
        return  new Properties();
    }
```

DatabaseConfiguration

```
@Bean
    public EntityManagerFactory entityManagerFactory() {
        HibernateJpaVendorAdapter vendorAdapter = new
HibernateJpaVendorAdapter();
        vendorAdapter.setGenerateDdl(true);
        LocalContainerEntityManagerFactoryBean factory =
                new LocalContainerEntityManagerFactoryBean();
        factory.setJpaDialect(new HibernateJpaDialect());
        factory.setDataSource(dataSource());
        factory.setJpaVendorAdapter(vendorAdapter);
        factory.setPackagesToScan("com.ch.jpa");
        factory.afterPropertiesSet();
        return factory.getObject();
    }
    @Bean
    public JpaTransactionManager transactionManager (EntityManagerFactory
entityManagerFactory) {
        return new JpaTransactionManager(entityManagerFactory);
    }
}
```

Dept

```
package com.ch.jpa.model;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Data;
@Entity
@Table(name="dept")
@Data
public class Dept {
    @Id
    private int deptno;
    private String dname;
    private String loc;
}
```

Emp

```
package com.ch.jpa.model;
import java.sql.Date;                          import javax.persistence.Column;
import javax.persistence.Entity;               import javax.persistence.Id;
import javax.persistence.JoinColumn;  import javax.persistence.ManyToOne;
import javax.persistence.Table;                import lombok.Data;
@Entity
@Table(name = "emp")
@Data
public class Emp {
    @Id // id가 primary key
    @Column(name = "empno")
    private int empno;
    private String ename;                  private String job;
    private Integer mgr;                   private Date hiredate;
    private int sal;                       private Integer comm;
    private int deptno;                   private String mgrName;
    @ManyToOne
    @JoinColumn(name =
"deptno",referencedColumnName="deptno",insertable=false, updatable=false)
    private Dept dept;
}
```

DeptDao

```
package com.ch.jpa.dao;
import java.util.List;
import javax.transaction.Transactional;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import com.ch.jpa.model.Dept;
@Repository
public interface DeptDao extends JpaRepository<Dept, Object>{
    @Query("select d from Dept d order by d.deptno")
    List<Dept> list();
    @Query("select d from Dept d where deptno=:deptno")
    Dept select(@Param("deptno") int deptno);
    public Dept saveAndFlush(Dept dept);
    @Transactional
    @Modifying
    @Query("delete from Dept d where d.deptno=:deptno")
    int delete(@Param("deptno") int deptno);
}
```

EmpDao

```
import javax.transaction.Transactional;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
@Repository
public interface EmpDao extends JpaRepository<Emp, Object> {
    @Query("select e, d from Emp e, Dept d where e.deptno=d.deptno")
    List<Emp> allList();
    @Query("select e from Emp e where deptno=:deptno")
    List<Emp> list(@Param("deptno") int deptno);
    @Query("select e from Emp e where empno=:empno")
    Emp select(@Param("empno") int empno);
    @SuppressWarnings("unchecked")
    public Emp saveAndFlush(Emp emp);
    @Query("select e from Emp e order by empno")
    List<Emp> list();
    @Transactional
    @Modifying
    @Query("delete from Emp where empno=:empno")
    int delete(@Param("empno") int empno);
}
```

DeptServiceImpl

```
@Service
public class DeptServiceImpl implements DeptService {
    @Autowired
    private DeptDao dd;
    public List<Dept> list() {
        return dd.list();
    }
    public Dept select(int deptno) {
        return dd.select(deptno);
    }
    public Dept insert(Dept dept) {
        return dd.saveAndFlush(dept);
    }
    public Dept update(Dept dept) {
        return dd.saveAndFlush(dept);
    }
    public int delete(int deptno) {
        return dd.delete(deptno);
    }
}
```

EmpServiceImpl

```
@Service
public class EmpServiceImpl implements EmpService {
    @Autowired
    private EmpDao ed;
    public List<Emp> list(int deptno) {     return ed.list(deptno);       }
    public Emp select(int empno) {          return ed.select(empno);   }
    public Emp insert(Emp emp) {
        return ed.saveAndFlush(emp);
    }
    public int update(Emp emp) {
        ed.saveAndFlush(emp);
        return 1;
    }
    public int delete(int empno) {          return ed.delete(empno);    }
    public List<Emp> allList() {
        return ed.allList();
    }
    public List<Emp> list() {
        return ed.list();
    }
}
```

Thymeleaf 기본 표현

- Simple expressions:
  - Variable Expressions: **${...}**
  - Selection Variable Expressions: ***{...}**
  - Message Expressions: **#{...}**
  - Link URL Expressions: **@{...}**

- Text operations:
  - String concatenation: **+**
  - Literal substitutions: **|The name is ${name}|**

- Arithmetic operations:
  - Binary operators: **+**, **-**, ***, */*, **%**
  - Minus sign (unary operator): **-**

- Boolean operations:
  - Binary operators: **and** , **or**
  - Boolean negation (unary operator): **!** , **not**

- Comparisons and equality:
  - Comparators: **>** , **<** , **>=** , **<=** (**gt** , **lt** , **ge** , **le**)
  - Equality operators: **==** , **!=** (**eq** , **ne**)

Thymeleaf 기본 표현

표현식 : th:[속성]="서버 전달 받은 값 또는 조건식"
Tag 안에 삽입되면 된다.
Thymeleaf 3.x 에서는 inline 표현식이 추가되어 html 태그없이 표현이 가능하다.

Title	Description
th:text	텍스트 내용
th:utext	html 내용
th:value	element value값, checkbox, input 등 <input type="text" th:value="${title}" />
th:with	변수값 지정 <p th:with="authType = ${user.authType}+' Type'" th:text="${authType}">
th:switch th:case	switch-case문 User is an administrator User is a manager
th:if	<p th:if="${user.authType}=='web'" th:text="${user.authType}">
th:unless	else
th:each	반복문 <p th:each="user : ${users}" th:text="${user.name}">

예제

```
@Controller
public class UserTestController {
    @GetMapping("/users")
    public String getUserList(Model model) {
        List<User> users = new ArrayList<>();
        for(int i=0;i<3;i++) {
            users.add(new User("kkaok"+i, "테스트"+i, "web") );
        }
        User user = new User("테스트ID", "테스터", "web") ;
        model.addAttribute("user", user);
        model.addAttribute("users", users);
        model.addAttribute("today", new Date());
        model.addAttribute("content", "<div><span style='font-size:20px'>Hello
World</span></div>");
        return "users";
    }
}
```

예제

```
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Thymeleaf 예제</title>
    <!-- Link 예제 -->
    <script th:src="@{/assets/vendor/jquery/jquery.js}"></script>
</head><body>
<h1>Collection 객체 each 예제</h1>
<p th:each="user : ${users}" th:text="${user.name}"></p>

<h1>Utility Objects 예제</h1>
<p>#calendars <br>Today is: <span th:text="${#calendars.format(today,'yyyy-MM-
dd')}">2019-02-15</span></p>
<p>#arrays <br>user count is <span th:text="${#arrays.length(users)}"></span>.</p>
 <h1>객체 예제</h1>
<p th:text="${user.name}">default</p>
<p>[[${user.name}]]</p>

<h1>th:text 예제</h1>
<p th:text="${content}">default value</p>

<h1>th:utext(Html) 예제</h1>
<p th:utext="${content}">default value</p>
```

예제

Collection 객체 each 예제
테스트0
테스트1
테스트2

Utility Objects 예제
#calendars
Today is: 2019-02-22

#arrays
user count is 3.

객체 예제
테스터

테스터

th:text 예제
Hello World

th:utext(Html) 예제
Hello World

예제

```
<h1>th:with 예제 : 변수 선언</h1>
<p th:with="authType = ${user.authType}+' Type'" th:text="${authType}"></p>

<h1>th:if 예제</h1>
<p th:if="${user.authType}=='web'" th:text="${user.authType}"></p>

<h1>th:unless 예제</h1>
<p th:unless="${user.authType}=='facebook'" th:text="'not '+
${user.authType}"></p>

<h1>script 예제</h1>
<p><span id="scriptTest"></span></p>
<script>
// script에서 값을 받을 때
var userId = '[[${user.userId}]]';
console.log(userId);
$("#scriptTest").html(userId);
</script>
</body>
</html>
```

예제

th:with 예제 : 변수 선언
web Type

th:if 예제
web

th:unless 예제
not web

script 예제
테스트ID

Thymeleaf 화면 구성(Layout)

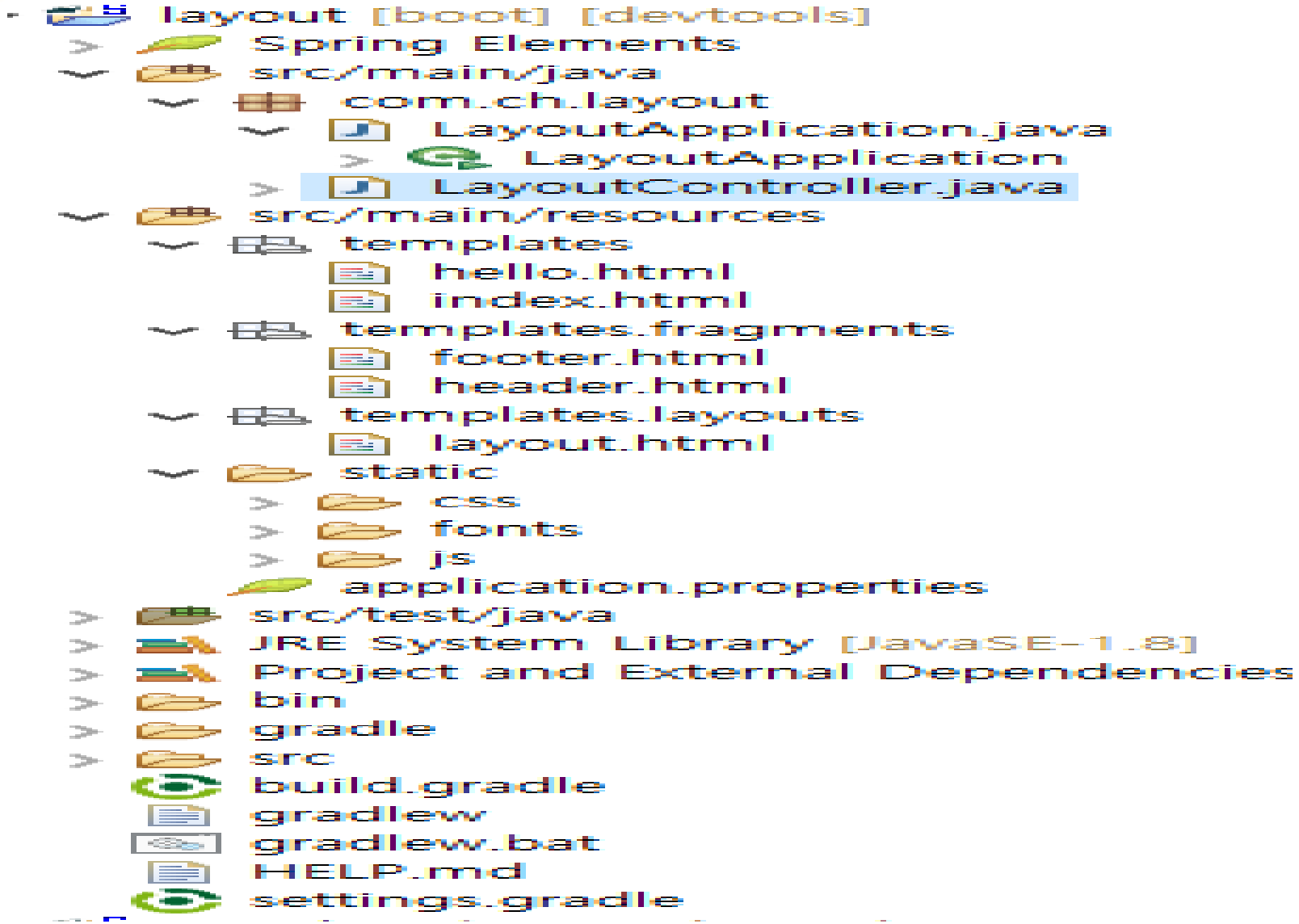
```
repositories {
        mavenCentral()
}
dependencies {
        implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
        implementation 'org.springframework.boot:spring-boot-starter-web‘

        implementation 'nz.net.ultraq.thymeleaf:thymeleaf-layout-dialect‘

        compileOnly 'org.projectlombok:lombok'
        developmentOnly 'org.springframework.boot:spring-boot-devtools'
        annotationProcessor 'org.projectlombok:lombok'
        testImplementation('org.springframework.boot:spring-boot-starter-test') {
                exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
        }
}
```

Thymeleaf 화면 구성(Layout)

thyme



LayoutController

```
package com.ch.layout;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class LayoutController {
		@RequestMapping("/index.do")
		public String frag(Model model) {
			return "index";
		}
		@RequestMapping("/hello.do")
		public String hello(Model model) {
			return "hello";
		}
}
```

Thymeleaf 화면 구성(Layout)

templates.fragments/footer.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<footer th:fragment="footerFragment">
    <div style="border: 1px solid gold">
        Footer영역입니다.
    </div>
</footer>
</html>
```

templates.fragments/header.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<header th:fragment="headerFragment">
    <div style="border: 1px solid green">
        header.html 입니다.
    </div>
</header>
</html>
```

templates.layoys/layout.html

```
<!DOCTYPE html>
<html lang="ko"
        xmlns:th="http://www.thymeleaf.org"
        xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
</head>
<body>
        <header th:replace="fragments/header :: headerFragment"></header>
        <div layout:fragment="content"></div>
        <footer th:replace="fragments/footer :: footerFragment"></footer>
</body>
</html>
```

templates/hello.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layouts/layout}">
<head>
</head>
<div layout:fragment="content">
    <div style="border: 1px solid red; height:300px;">
            hello.html 입니다.
            즉 컨텐츠 영역
    </div>
</div>
</html>
```

templates/index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layouts/layout}">
<head>
<!--    thymeleaf layout dialect가 contents페이지의 head태그를 layout의 head태
그에 자동으로 추가해준다. -->
<!--    필요힌 css, script 추가영역 -->
</head>
<div layout:fragment="content">
    <div style="border: 1px solid red; height:300px;">
        index.html 입니다.
        즉 컨텐츠 영역
    </div>
</div>
</html>
```

th:insert, th:replace, th:include 차이 비교

```
<!-- th:include -->
   <th:block th:include="fragments/commonHead"></th:block>
   <!-- th:insert -->
   <th:block th:insert="fragments/commonHead"></th:block>
   <!-- th:replace -->
   <th:block th:replace="fragments/commonHead"></th:block>
```

위의 세가지는 동일한 결과물을 만듭니다. 주요 차이점은 fragment의 태그를 포함하느냐 안하느냐 인데 위의 예제에서는 태그를 <th:block>을 이용하였고 commonHead.html 자체도 tag가 없이 페이지 전체를 include하기 때문에 의미가 없습니다.

태그와 fragment를 사용하는 예제

```
   <footer th:include="fragments/commonFooter :: commonFooter"></footer>
   <footer th:replace="fragments/commonFooter :: commonFooter"></footer>
   <footer th:insert="fragments/commonFooter :: commonFooter"></footer>
```

결과 화면

```
   <footer>       Layout Common Footer    </footer>
   <footer>       Layout Common Footer    </footer>
   <footer><footer>       Layout Common Footer    </footer></footer>
```

th:include와 th:replace는 fragment 태그를 제외하고 결과를 가져오고 th:insert는 fragment 태그를 포함해서 가져오는 차이가 있습니다.

태그 제외

<footer th:include="fragments/commonFooter :: commonFooter"
th:remove="tag"></footer>
<footer th:replace="fragments/commonFooter :: commonFooter"
th:remove="tag"></footer>
<footer th:insert="fragments/commonFooter :: commonFooter"
th:remove="tag"></footer>
th:remove="tag" attribute를 이용하는 테스트 입니다.

결과 화면
Layout Common Footer
<footer>
Layout Common Footer
</footer>
<footer>
Layout Common Footer
</footer>
th:include는 th:remove="tag"가 적용이 되서 태그가 사라졌습니다.

결론적으로 공통을 처리하는 것은 th:include, th:replace, th:insert가 있고 tag를 제외 하고 싶다면 태그 자체를 <th:block></th:block>를 이용하는게 좋습니

Layout 사용하기

레이아웃 선택하기

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
  xmlns:layout="http://www.ultraq.net.nz/web/thymeleaf/layout"
  layout:decorator="layout/defaultLayout">
</html>
```

layout:decorator에 선언을 합니다. 이때 확장자는 제외합니다.

컨텐츠 구성

```
<th:block layout:fragment="content">
<h1>Page Content</h1>
</th:block>
<th:block layout:fragment="pageCustomScript">
    <script th:src="@{/assets/vendor/jquery/jquery.js}"></script>
</th:block>
<th:block layout:fragment="pageCustomCss">
    <script th:src="@{/assets/vendor/jquery/jquery.css}"></script>
</th:block>
```

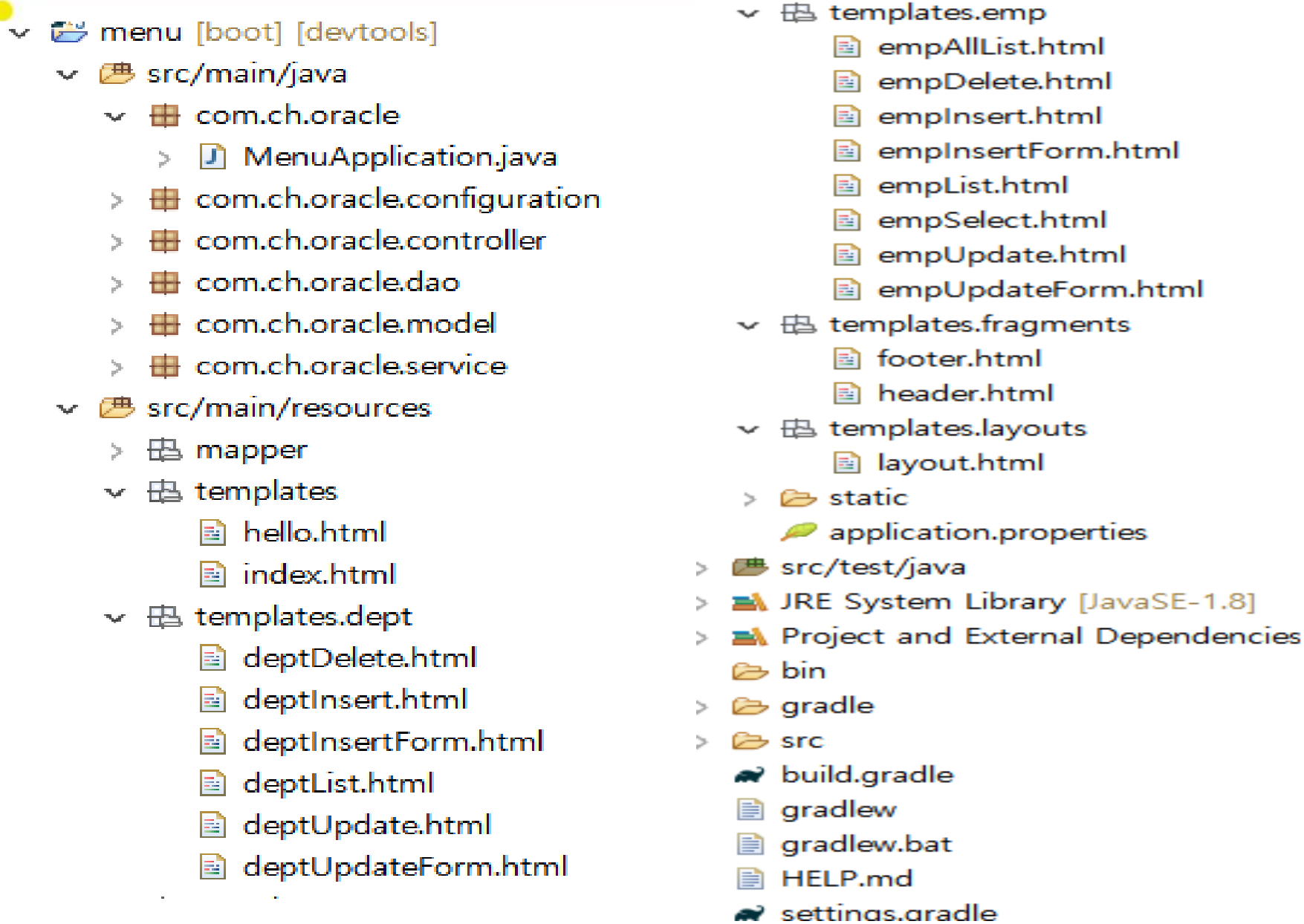
layout:fragment를 이용하여 구현하면 Layout 페이지에 선언된 위치에 컨텐츠가 생성이 됩니다.

위의 예제에는 <th:block>을 이용했지만 <section>, , <p>, <footer> 등 원하는 태그를 사용해서 구현할 수 있습니다.

Layout 프로젝트

```
repositories {
        mavenCentral()
}
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'nz.net.ultraq.thymeleaf:thymeleaf-layout-dialect'
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.oracle.database.jdbc:ojdbc8'
    annotationProcessor 'org.springframework.boot:spring-boot-configuration-
processor'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}
test {
        useJUnitPlatform()
}
```

Layout 프로젝트


menu [boot] [devtools]
src/main/java
com.ch.oracle
MenuApplication.java
com.ch.oracle.configuration
com.ch.oracle.controller
com.ch.oracle.dao
com.ch.oracle.model
com.ch.oracle.service
src/main/resources
mapper
templates
hello.html
index.html
templates.dept
deptDelete.html
deptInsert.html
deptInsertForm.html
deptList.html
deptUpdate.html
deptUpdateForm.html
templates.emp
empAllList.html
empDelete.html
empInsert.html
empInsertForm.html
empList.html
empSelect.html
empUpdate.html
empUpdateForm.html
templates.fragments
footer.html
header.html
templates.layouts
layout.html
static
application.properties
src/test/java
JRE System Library [JavaSE-1.8]
Project and External Dependencies
bin
gradle
src
build.gradle
gradlew
gradlew.bat
HELP.md
settings.gradle

templates.layouts/layout.html

```
<!DOCTYPE html>
<html lang="ko"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
</head>
<body>
      <header th:replace="fragments/header ::
headerFragment"></header>
      <div layout:fragment="content"></div>
      <footer th:replace="fragments/footer ::
footerFragment"></footer>
</body>
</htm
```

templates.fragments/header.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<header th:fragment="headerFragment">
    <div style="border: 1px solid green">
        <a class="btn btn-success" href="/dept/deptList.do">부서목록</a>
        <a class="btn btn-info" href="/dept/deptInsertForm.do">부서입력</a>
        <a class="btn btn-default" href="/emp/empAllList.do">전 직원 조회</a>
        <a class="btn btn-warning" href="/index">index</a>
        <a class="btn btn-danger" href="/hello">hello</a>
    </div>
</header>
</html>
```

templates.fragments/footer.html

```
<!DOCTYPE html><html xmlns:th="http://www.thymeleaf.org">
<footer th:fragment="footerFragment">
    <div style="border: 1px solid gold">
        Footer영역입니다.
    </div>
</footer>
</html>
```

templates.fragments/hello.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layouts/layout}">
<head>
</head>
<div layout:fragment="content">
    <div style="border: 1px solid red; height: 300px;" >
            hello.html 입니다.
            즉 컨텐츠 영역
    </div>
</div>
</html>
```

templates.fragments/index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layouts/layout}">
<head>
<!--    thymeleaf layout dialect가 contents페이지의 head태그를 layout의 head태그에 자동으로 추가해준다. -->
<!--    필요힌 css, script 추가영역 -->
</head>
<div layout:fragment="content">
    <div style="border: 1px solid red; height: 300px;">
        index.html 입니다.
        즉 컨텐츠 영역
    </div>
</div>
</html>
```

templates.fragments/deptList.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
    layout:decorate="~{layouts/layout}">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style></head><body>
<div layout:fragment="content">
<table class="table table-bordered">
        <caption class="text-primary">부서 목록</caption>
        <tr><th>부서코드</th><th>부서명</th><th>근무지</th>
            <th>수정</th><th>삭제</th></tr>
<tr th:if="${#lists.size(list)} == 0">
        <th colspan="5">부서 데이터가 없습니다</th></tr></th:if>
```

templates.fragments/deptList.html

```
<tr th:if="${#lists.size(list)} > 0" th:each="dept:${list}">
        <td th:text="${dept.deptno}"></td>
        <td><a href="/emp/empList.do?deptno="
                        th:attrappend="href=${dept.deptno}"
                        th:text=${dept.dname}
                class="btn btn-success btn-sm"></a></td>
        <td th:text="${dept.loc}"></td>
        <td><a class="btn btn-sm btn-warning"
                href="/dept/deptUpdateForm.do?deptno=${dept.deptno}">수정
</a></td>
                <!-- th:attrappend="href=${dept.deptno}" -->
        <td><a class="btn btn-sm btn-danger"
                href="/dept/deptDelete.do?deptno="
                th:attrappend="href=${dept.deptno}">삭제</a></td>
        </tr></th:if>
</table>
<a class="btn btn-info" href="/dept/deptInsertForm.do">부서입력</a>
<a class="btn btn-default" href="/emp/empAllList.do">전 직원 조회</a>
</div>
</body>
</html>
```

templates.fragments/insertForm.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layouts/layout}">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style>
<script type="text/javascript">
            function deptnoChk() {
                        if (!frm.deptno.value) {
                                    alert("부서코드 입력한 후에 체크하시오");
                                    frm.deptno.focus(); return false;
                        }
                        $.post("/dept/deptnoChk.do","deptno="+frm.deptno.value,
                                                function(data) {
                                    $('#disp').html(data);
                        });
            }
</script></head><body>
```

templates.fragments/insertForm.html

```
<div layout:fragment="content">
<h2>부서 정보 입력</h2>
<form action="/dept/deptInsert.do" method="post" name="frm">
<table class="table table-bordered">
        <tr><td>부서코드</td><td><input type="number" name="deptno"
                required="required" autofocus="autofocus">
                <input type="button" value="중복체크" onclick="deptnoChk()"
                class="btn btn-warning btn-sm">
                <div class="err" id="disp"></div></td></tr>
        <tr><td>부서명</td><td><input type="text" name="dname"
                required="required"></td></tr>
        <tr><td>근무지</td><td><input type="text" name="loc"
                required="required"></td></tr>
        <tr><th colspan="2"><input type="submit"></th></tr>
</table>
</form>
<a class="btn btn-info" href="deptList.do">부서목록</a>
</div>
</body>
</html>
```

templates.fragments/empAllList.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layouts/layout}">
<head><meta charset="UTF-8"><title>Insert title here</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
<script th:src="@{/js/jquery.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<style> .err { color: red; font-weight: bold; }</style></head><body>
<div layout:fragment="content">
<!-- <div class="container"> -->
<h2 class="text-primary">전 직원 목록</h2>
<table class="table table-striped">
            <tr><td>사번</td><td>이름</td><td>업무</td><td>입사일</td>
                        <td>급여</td><td>부서명</td><td>부서코드</td></tr>
```

templates.fragments/empAllList.html

```
<tr th:if="${#lists.size(list)} ==0">
		<th colspan="7">직원이 없습니다</th></tr></th:if>
<tr th:if="${#lists.size(list)} > 0" th:each="emp:${list}">
		<td th:text="${emp.empno}"></td>
		<td><a class="btn btn-sm btn-success" th:text="${emp.ename}"
			href="/emp/empSelect.do?empno=${emp.empno}" ></a></td>
		<!--	th:attrappend="href="></a></td> -->
		<td th:text="${emp.job}"></td>
		<td th:text="${emp.hiredate}"></td>
		<td th:text="${emp.sal}"></td> a
		<td th:text="${emp.dname}"></td>
		<td th:text="${emp.loc}"></td>
</table>
		<a class="btn btn-info" href="/dept/deptList.do">부서 목록</a>
</div></body></html>
```