Ch09 관리자 회원가입 기능 만들기

1. 전자 도서관 서비스 설계
2. 전자 도서관 서비스 프로젝트 생성하기
3. 관리자 회원가입 기능 구현
4. 데이터베이스 만들고 연동하기

Section 01

전자 도서관 서비스 설계

1. 전자 도서관 서비스의 관리자 기능 정의

■ 전자 도서관 서비스란?

- 학교 도서관, 시립 도서관 등에서 사용할 수 있는 프로그램
  - ✓ 주요 기능: 도서 대여, 반납, 희망 도서 요청, 회원가입, 회원 관리 등

■ 진행 내용

① 웹 서비스를 구현하기 위한 프로젝트의 전체적인 흐름과 구조를 이해한다.

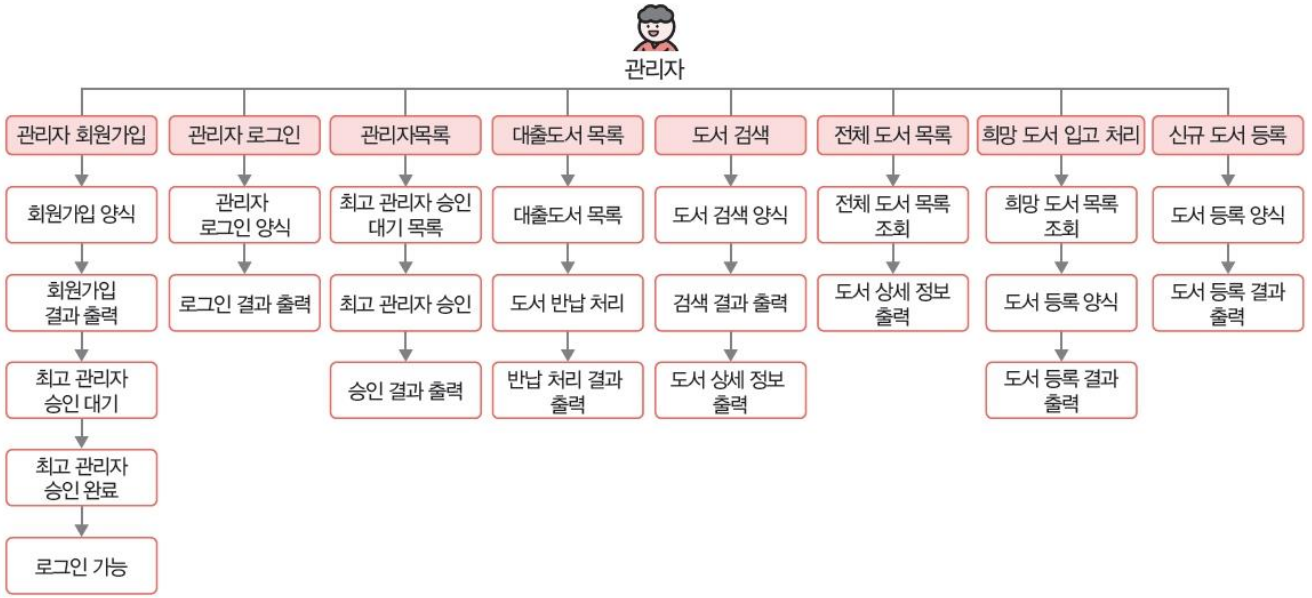
② 프로젝트를 생성하고 필요한 파일을 복사한다.

③ 관리자 회원가입 기능을 구현한다.

④ 데이터베이스 연동에 필요한 설정과 연동 방법을 학습한다.

⑤ 데이터 암호화 방법에 대해서 학습한다.

■ 도서관 서비스의 전체적인 기능 중에서 관리자에게 필요한 기능

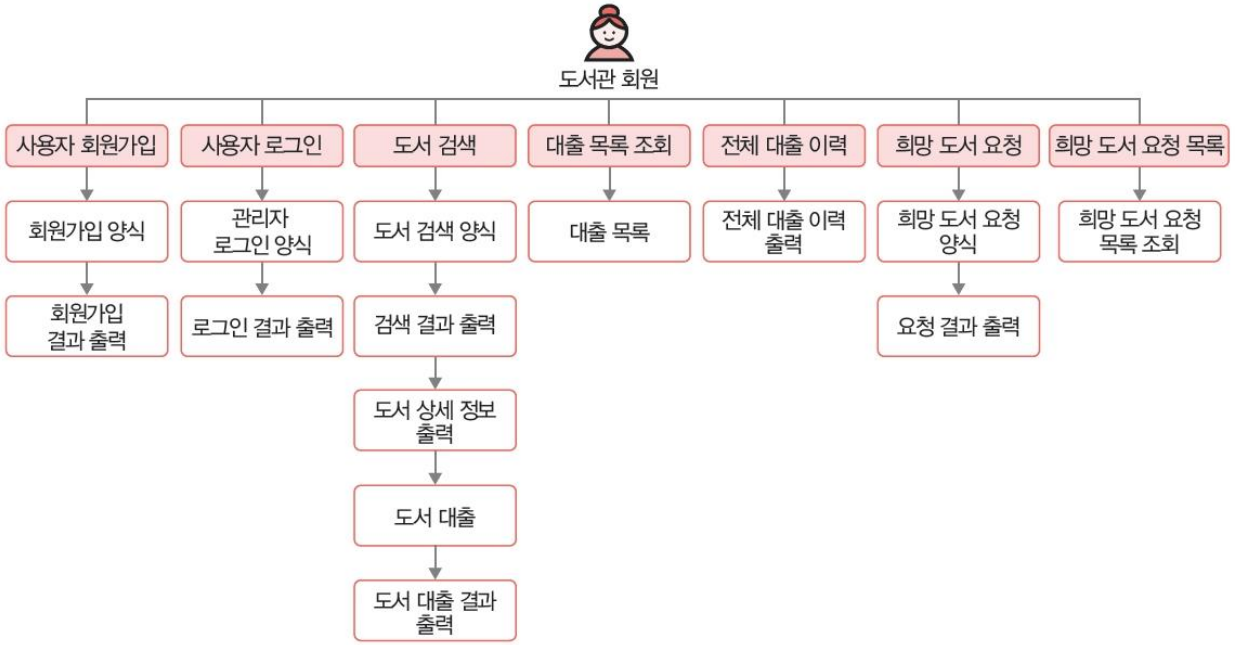


[그림 1]전자 도서관 서비스 관리자의 주요 기능

[표 1] 관리자 주요 기능 정의

관리자 주요 기능 정의		
기능	내용	학습 챕터
회원가입	관리자 회원가입 양식을 이용해서 회원가입이 가능하다. 회원가입 후 최고 관리자(super admin)의 승인이 이루어진 후 로그인이 가능하다.	9장, 10장
신규 도서 등록	새로 들어온 도서를 시스템에 등록할 수 있다.	13장
도서 반납	전체 대출 목록을 조회할 수 있으며, 특정 도서에 대해서 반납 처리할 수 있다.	14장
희망 도서 관리	사용자가 요청한 희망 도서를 새로 들어온 도서로 시스템에 등록할 수 있다.	11장

■ 도서관 서비스의 전체적인 기능 중에서 관리자에게 필요한 기능



[그림 2] 전자 도서관 서비스 사용자의 주요 기능

[표 2] 사용자 주요 기능 정의

사용자 주요 기능 정의		
기능	내용	학습 챕터
회원가입	사용자 회원가입 양식을 이용해서 회원가입이 가능하다. 회원가입 후 도서 대출 서비스 및 희망 도서 요청 서비스를 이용할 수 있다.	12장
도서 대출	특정 도서 검색 후 도서 대출이 가능하다면 대출할 수 있다.	13장
대출 이력 조회	사용자가 대출한 모든 이력을 조회할 수 있다.	13장
희망 도서 요청	사용자가 희망 도서를 요청할 수 있고, 관리자는 이를 입고 처리할 수 있다.	14장

Section 02

전자 도서관 서비스 프로젝트 생성하기

■ 프로젝트 생성하고 설정하기

- 1. STS의 Project Explorer에서

❶ 오른쪽 마우스를 클릭해서 [New]-[Project...]를 클릭하기.

❷ [New Project] 창에서 'spring'을 검색하고,

❸ 'Spring Legacy Project'를 선택한 후

❹ <Next> 버튼을 클릭하기. 이어서

❺ [New Spring Legacy Project] 창의 Project name에 BookRentalPjt를 입력하고

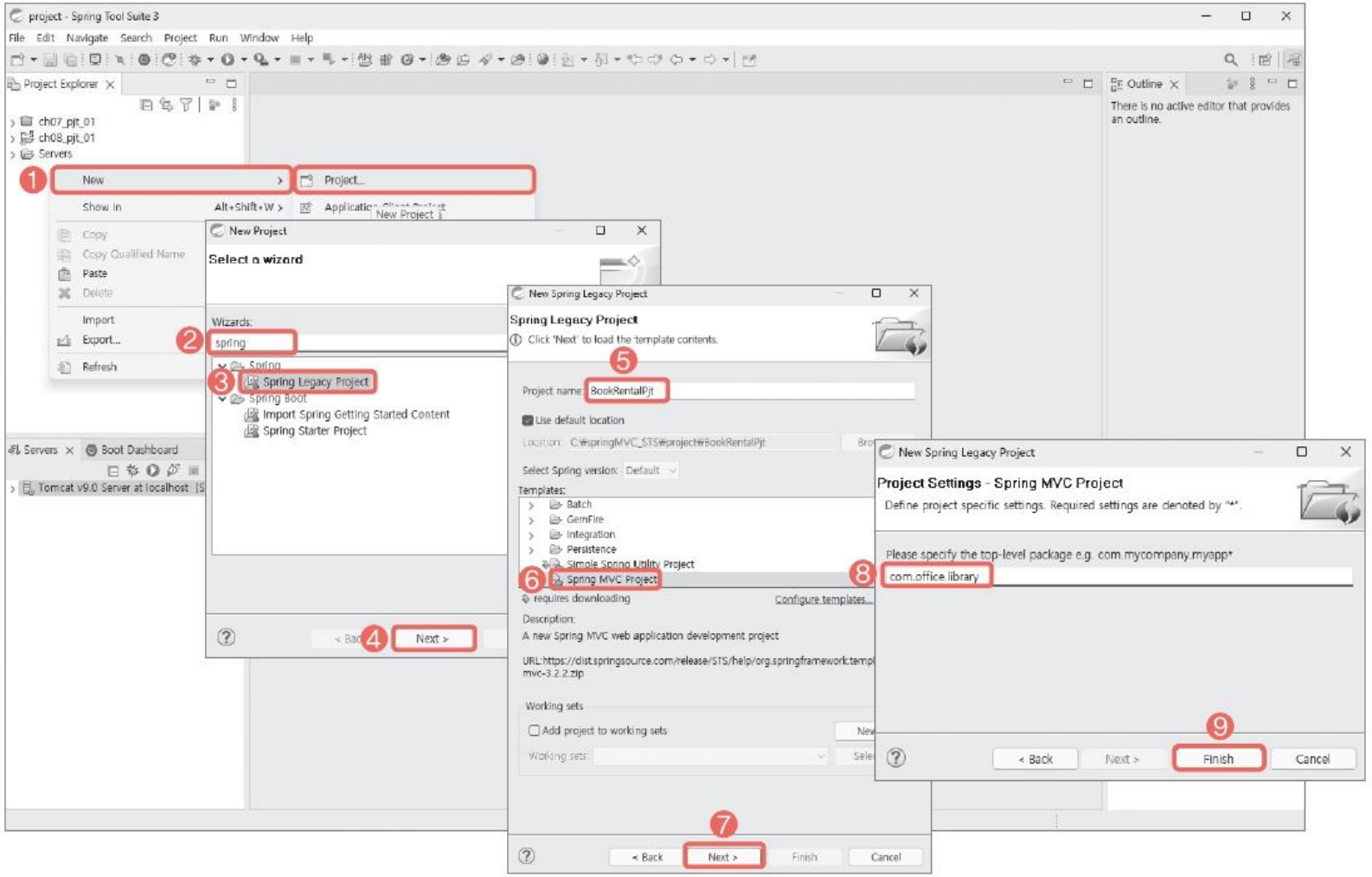
❻ Template 목록에서 'Spring MVC Project'를 선택한 후,

❼ <Next> 버튼을 클릭하기. 이어서

❽ 패키지 이름으로 com.office.library를 입력하고

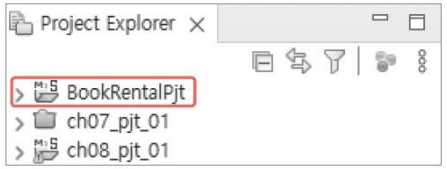
❾ <Finish> 버튼을 클릭함

■ 프로젝트 생성하고 설정하기



■ 프로젝트 생성하고 설정하기

- 2. BookRentalPjt 프로젝트가 정상적으로 생성된 것 확인하기



- 3. pom.xml 파일에서 자바와 스프링 버전을 수정하고 파일을 저장하기

```
<properties>
    <java-version>11</java-version>
    <org.springframework-version>5.2.9.RELEASE</org.springframework-version>
    ...생략...
</properties>

...생략...

<configuration>
    <source>11</source>
    <target>11</target>
    <compilerArgument>-Xlint:all</compilerArgument>
    <showWarnings>true</showWarnings>
    <showDeprecation>true</showDeprecation>
</configuration>
```

■ 프로젝트 생성하고 설정하기

- 4. pom.xml의 수정 내용이 반영될 수 있도록 프로젝트를 업데이트하기
  - ✓ BookRentalPjt 프로젝트에서 마우스 오른쪽 버튼을 클릭하여 [Maven]-[Update Project...]를 클릭하여 프로젝트를 업데이트하기
  - ✓ 프로젝트가 정상적으로 업데이트되면 JRE 버전이 6에서 11로 변경됨

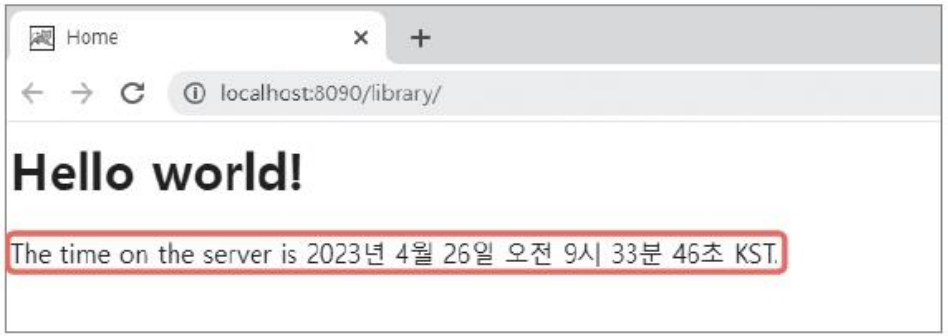
■ 프로젝트 생성하고 설정하기

- 5. 한글이 깨지는 문제를 해결하기 위해 web.xml에 <filter>를 추가하기

```
...생략...
</servlet-mapping>

<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
        <param-name>forceEncoding</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

</web-app>
```

■ 프로젝트 생성하고 설정하기

- 6. 프로젝트 생성을 모두 마쳤다면 프로젝트를 톰캣에서 실행하기

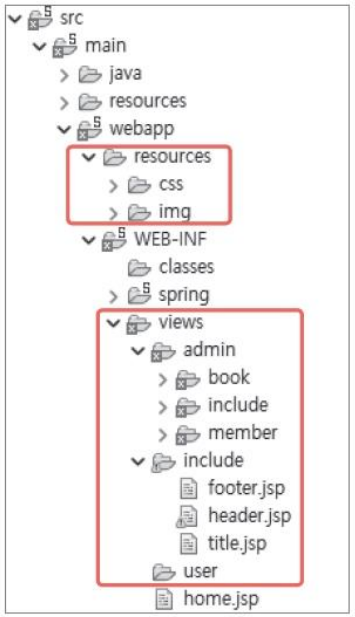


■ 웹 문서 제작에 필요한 파일 생성하기

- 만약 톰캣이 실행 중이라면 중지하고 다음 과정을 따라 하기
- 1. [views] 폴더에 [admin], [user], [include] 폴더를 각각 생성함
  - ✓ [views] 폴더 위치: src/main/webapp/WEB-INF/views
  - ✓ [admin] 폴더: 관리자 관련 페이지가 들어 있는 폴더
  - ✓ [include] 폴더: 관리자와 사용자가 공통으로 사용하는 페이지가 들어 있는 폴더
  - ✓ [user] 폴더: 사용자 관련 페이지가 들어 있는 폴더

■ 웹 문서 제작에 필요한 파일 생성하기

- 2. [admin], [include] 폴더 준비 (제공 소스 폴더 [Chap09]-[Book RentalPjt_준비]-[view] 폴더 이용)
  - ✓ [admin] 폴더: [book], [include], [member] 폴더를 붙여넣기
  - ✓ [include] 폴더: footer.jsp, header.jsp, title.jsp 파일을 붙여넣기
  - ✓ [user] 폴더는 사용자 관련 페이지로, 지금은 관리자 관련 기능을 구현하므로 비워두기
- include 또는 import해야 하는 파일이 없어 폴더와 파일에 'X' 표시 생김
  - ✓ 우선은 무시하고 계속 진행
- 3. [resources] 폴더: [css], [img] 폴더를 붙여넣기



[그림 3] BookRentalPjt의 최종 파일 구조

Section 03

관리자 회원가입 기능 구현

■ 패키지와 클래스를 생성

- 관리자 회원가입 기능도 Controller, Service, DAO 객체를 이용해서 구현함
- [src/main/java]에 다음 패키지와 클래스 생성하기

com.office.library.admin 패키지	com.office.library.admin.member 패키지
AdminHomeController.java	• Controller 클래스: AdminMemberController.java • Dao 클래스: AdminMemberDao.java • Service 클래스: AdminMemberService.java • Vo 클래스: AdminMemberVo.java

[그림 4] 생성된 관리자 회원가입 패키지와 클래스

■ 관리자 홈 화면

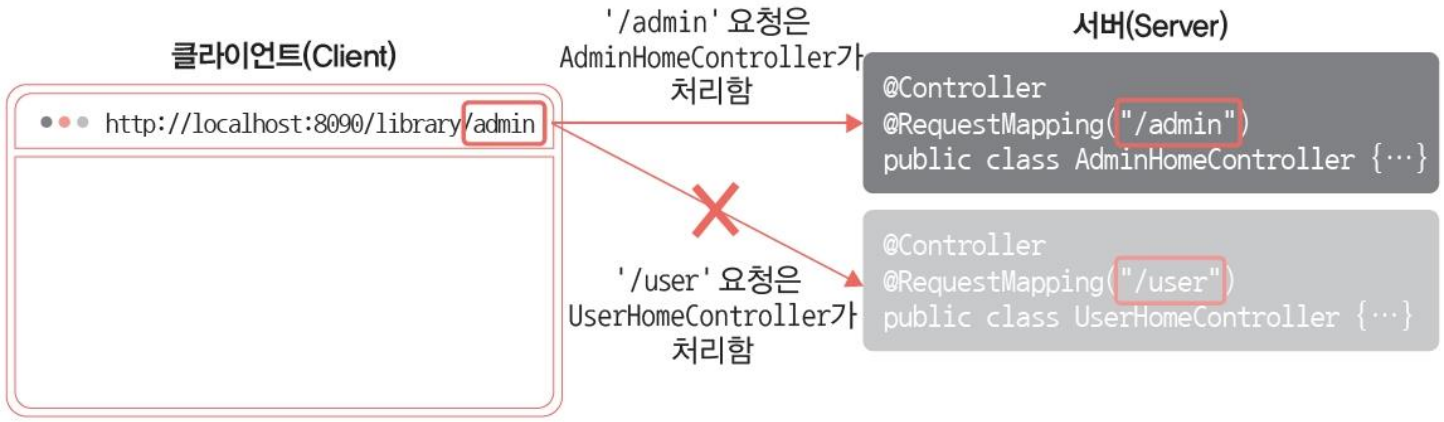
- 관리자 홈에 접속하기 위한 URL : http://localhost:8090/library/admin
  - ✓ AdminHomeController에 이를 처리할 수 있는 메서드를 만들고 /admin을 매핑함
- AdminHomeController에 home() 메서드를 다음과 같이 코딩하기

코드 9-1 BookRentalPjt\src\main\java\com\office\library\admin\AdminHomeController.java

```
01 package com.office.library.admin;
02
03 import org.springframework.stereotype.Controller;
04 import org.springframework.web.bind.annotation.RequestMapping;
05 import org.springframework.web.bind.annotation.RequestMethod;
06
07 @Controller
08 @RequestMapping("/admin")
09 public class AdminHomeController {
10
11     @RequestMapping(value = {"", "/"}, method = RequestMethod.GET)
12     public String home() {
13         System.out.println("[AdminHomeController] home()");
14
15         String nextPage = "admin/home";
16
17         return nextPage;
18     }
19 }
```

■ 관리자 홈 화면

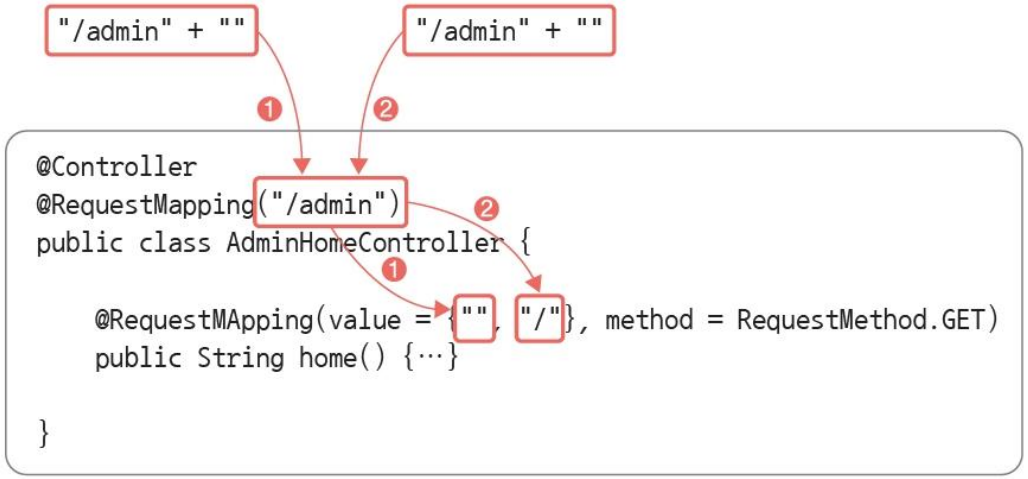
- AdminHomeController
  - ✓ @Controller에 의해서 프로젝트가 실행될 때 스프링 IoC 컨테이너에 빈 객체로 생성됨
  - ✓ @RequestMapping("/admin")을 명시함으로써 기본적으로 /admin에 대한 요청을 AdminHomeController가 처리하게 됨
  - ✓ 이때 @RequestMapping("/admin")은 필수가 아님
  - ✓ 만약 @RequestMapping("/admin")을 명시하지 않으면 AdminHomeController는 '/admin' 요청뿐만 아니라 클라이언트의 모든 요청을 처리할 수 있는 컨트롤러가 됨



[그림 5] 클라이언트의 '/admin' 요청을 처리하는 AdminHomeController

■ 관리자 홈 화면

- AdminHomeController
  - ✓ home()에 매핑되어 있는 문자열: "", "/"
  - ✓ " /admin " 과 " /admin/ " 요청을 모두 home()이 처리할 수 있게 됨



[그림 6] 매핑 데이터가 여러 개인 경우

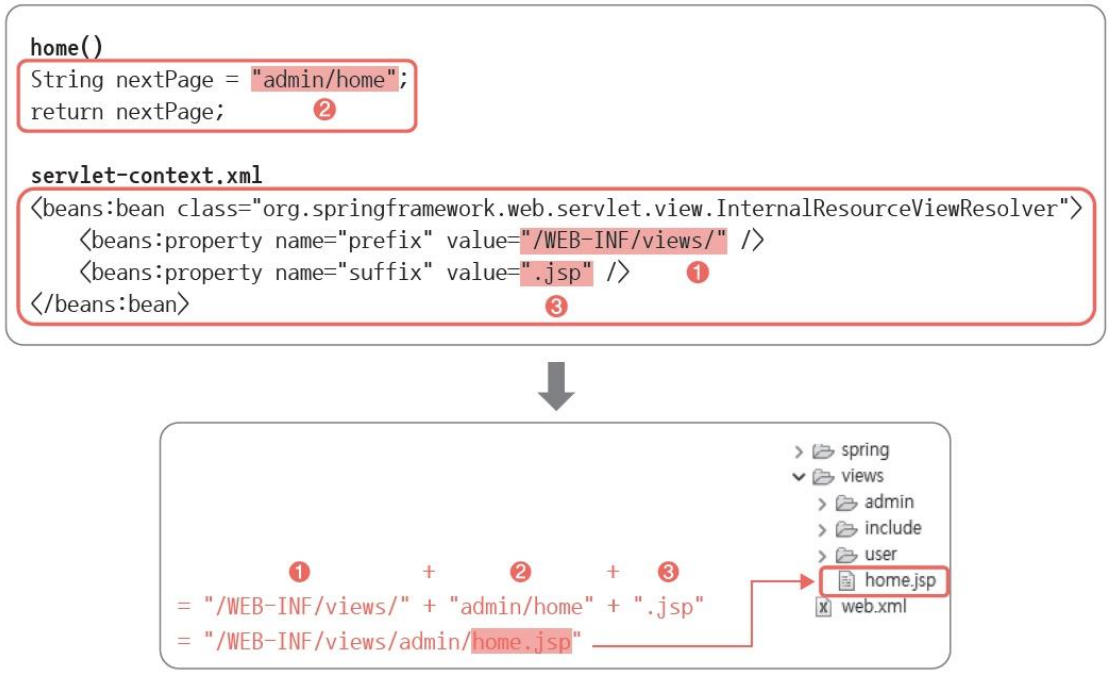
■ 관리자 홈 화면

- @RequestMapping
  - ✓ value와 method 속성을 가짐
  - ✓ value 속성: 사용자 요청 정보를 명시함
  - ✓ method 속성: 요청 정보가 전송되는 방식(get, post)을 명시함(기본값: get)
  - ✓ value 속성만 있다면 method를 명시하지 않고 value 속성 값만 명시할 수도 있음
  - ✓ 다음 ❶~❸번은 모두 동일한 매핑임

```
❶ @RequestMapping(value = {"", "/"}, method = RequestMethod.GET)
❷ @RequestMapping(value = {"", "/"})
❸ @RequestMapping({"", "/"})
```

■ 관리자 홈 화면

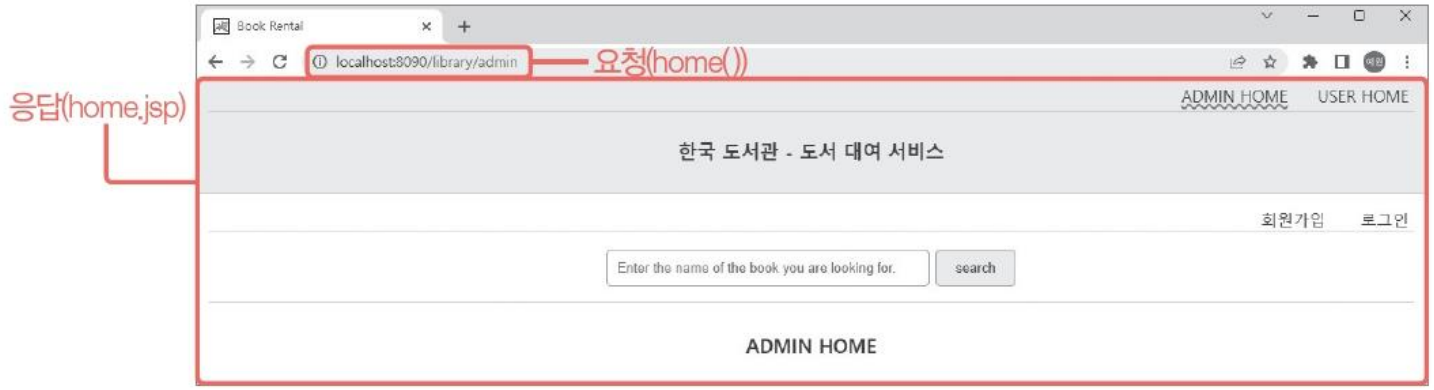
- home()
  - ✓ "admin/home"을 반환함
  - ✓ 이에 따르면 [views]의 [admin] 폴더에 있는 home.jsp를 이용해서 뷰로 만들고 이를 이용해서 클라이언트에 응답함



[그림 7] 사용자 요청에 응답하기 위한 JSP 파일 경로 설정

■ 관리자 홈 화면

- 톰캣에서 프로젝트를 실행 후 /admin을 서버에 요청하고 응답 화면(home.jsp)을 확인하기
  - ✓ 요청 URL: http://localhost:8090/library/admin



[그림 8] 실행된 home.jsp 화면

■ 관리자 회원가입 화면

- admin/home.jsp가 nav.jsp를 include하는 코드

```
<jsp:include page="./include/nav.jsp" />
```

- nav.jsp
  - ✓ 관리자 화면의 전체적인 메뉴로 '회원가입'을 클릭하면 서버에 /admin/member/createAccountForm 요청이 발생gka

```
<a href="<c:url value='/admin/member/createAccountForm' />">회원가입</a>
```

nav.jsp에 메뉴( )가 2번 보이는데, 어떤 차이가 있나요?

메뉴는 로그인 전/후로 구분해서 보여줄 필요가 있습니다. 로그인 상태는 세션 session을 이용하는데요. 아직 로그인 기능과 세션에 대해서 학습하지 않았기 때문에 지금은 '관리자 로그인 전 메뉴'만 보면 됩니다.

■ 관리자 로그인 후 메뉴

```
<div class="menu">
    <ul>
        <li><a href="<c:url value='/admin/member/logoutConfirm' />">로그아웃</a></li>
        <li><a href="<c:url value='/admin/member/modifyAccountForm' />">계정수정</a></li>
        ...생략...
    </ul>
</div>
```

■ 관리자 로그인 전 메뉴

```
<div class="menu">
    <ul>
        <li><a href="<c:url value='/admin/member/loginForm' />">로그인</a></li>
        <li><a href="<c:url value='/admin/member/createAccountForm' />">회원가입</a></li>
    </ul>
</div>
```

AdminMemberController

- /createAccountForm 요청을 처리함
  - ✓ com.office.library.admin.member 패키지의 AdminMemberController.java에 @Controller, @RequestMapping을 명시해서 AdminMemberController가 컨트롤러의 역할을 할 수 있도록 함

코드 9-2 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberController.java

```
01 package com.office.library.admin.member;
02
03 import org.springframework.stereotype.Controller;
04 import org.springframework.web.bind.annotation.RequestMapping;
05
06 @Controller
07 @RequestMapping("/admin/member")
08 public class AdminMemberController {
09
10 }
```

createAccountForm()

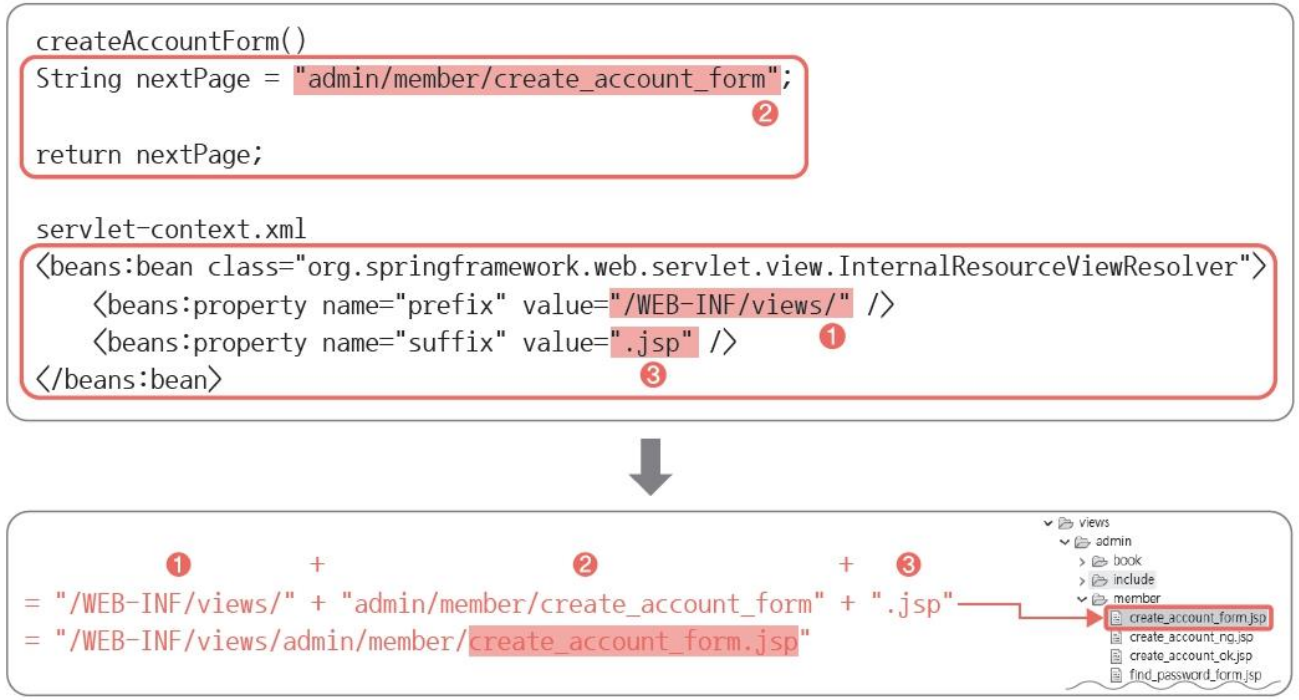
- /createAccountForm 요청을 처리할 수 있는 메서드
  - ✓ 메서드 이름은 편의상 요청 정보와 동일하게 설정함
  - ✓ 특별히 할 일은 없고, 관리자가 회원가입을 할 수 있도록 회원가입 양식이 있는 화면을 응답해주기만 하면 됨
  - ✓ 따라서 createAccountForm()에 회원가입 양식이 있는 페이지를 반환하는 코드 넣기

코드 9-3 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberController.java

```
...생략...
@Controller
@RequestMapping("/admin/member")
public class AdminMemberController {

    // 회원가입
    @RequestMapping(value = "/createAccountForm", method = RequestMethod.GET)
    public String createAccountForm() {
        System.out.println("[AdminMemberController] createAccountForm()");
        String nextPage = "admin/member/create_account_form";

        return nextPage;
    }
}
```

createAccountForm()

- 반환값 admin/member/create_account_form은 home.jsp와 마찬가지로 InternalResourceViewResolver에 의해 create_account_form.jsp를 클라이언트에 반환함



[그림 9] creat_account_form.jsp 파일 경로 설정

create_account_form.jsp

- 관리자 회원가입 페이지: <form>을 이용해서 관리자 정보를 입력할 수 있음

코드 9-4 BookRentalPjt\src\main\webapp\WEB-INF\views\admin\member\create_account_form.jsp

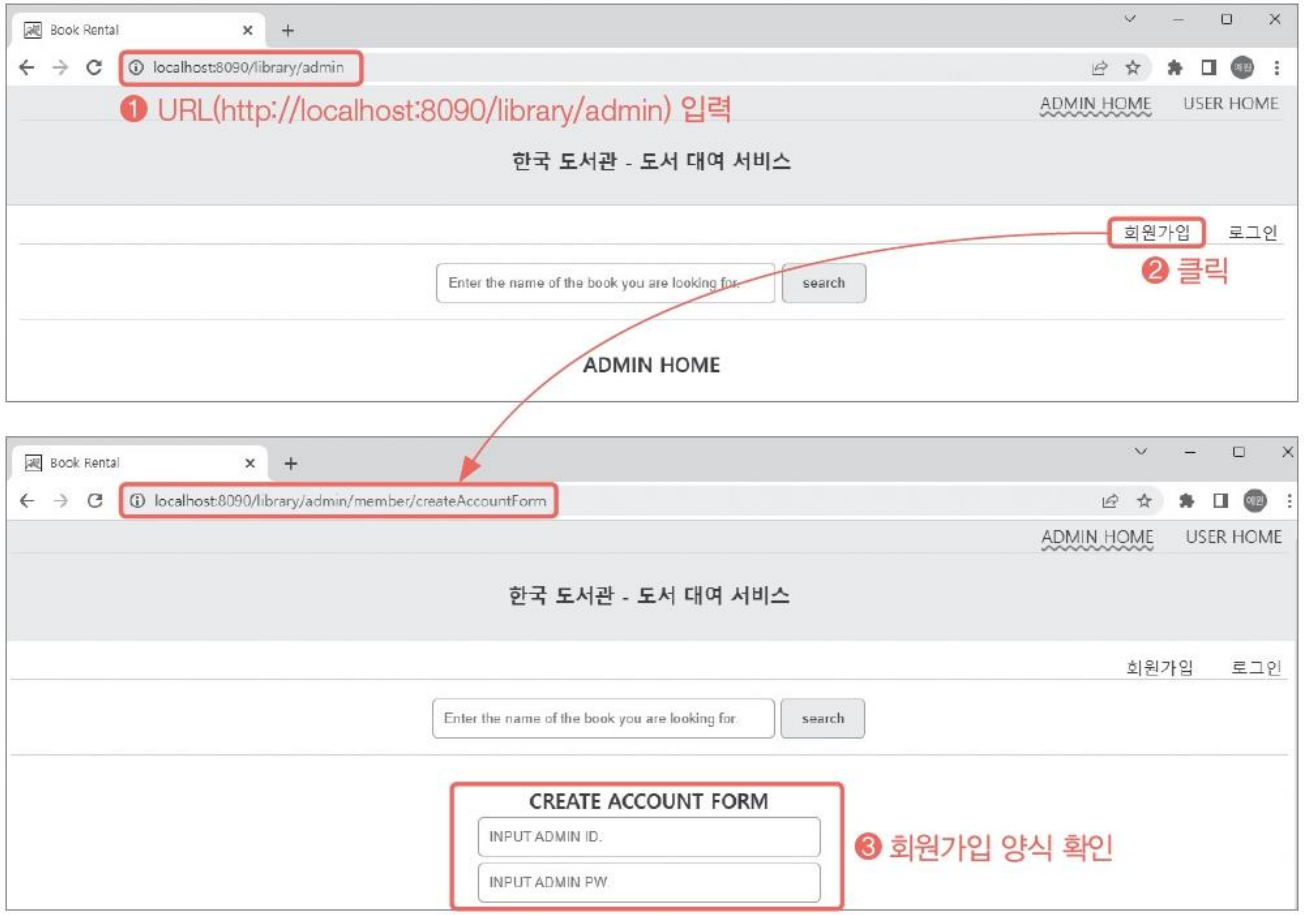
```
01 ...생략...
02
03 <div class="create_account_form">
04
05 <form action="<c:url value='/admin/member/createAccountConfirm' />"
       name="create_account_form" method="post">
06
07 <input type="text"name="a_m_id" placeholder="INPUT ADMIN ID."> <br>
08 <input type="password"name="a_m_pw" placeholder="INPUT ADMIN PW."> <br>
09 <input type="password"name="a_m_pw_again"placeholder="INPUT ADMIN PW AGAIN."> <br>
10 <input type="text"name="a_m_name" placeholder="INPUT ADMIN NAME."> <br>
11 <select name="a_m_gender">
12     <option value="">SELECET ADMIN GENDER.</option>
13     <option value="M">Man</option>
14     <option value="W">Woman</option>
15 </select> <br>
16 <input type="text"name="a_m_part" placeholder="INPUT ADMIN PART."> <br>
17 <input type="text"name="a_m_position" placeholder="INPUT ADMIN POSITION."> <br>
18 <input type="email"name="a_m_mail"placeholder="INPUT ADMIN MAIL." ><br>
19 <input type="text"name="a_m_phone"placeholder="INPUT ADMIN PHONE."> <br>
20 <input type="button"value="create account" onclick="createAccountForm();">
21 <input type="reset"value="reset">
22
23 </form>
24 </div>
25 ...생략...
```

■ name 속성

- <input>과 <select>의 name에 명시된 속성 값은 실제로 관리자 정보가 저장되는 데이터베이스의 컬럼명과 동일하게 함
  - ✓ 반드시 name을 컬럼명과 같아야 하는 것은 아니지만, name과 컬럼명이 같다면 더 편리하기 때문에 대체로 동일하게 설정함

■ 실행 결과

- 톰캣에서 프로젝트를 실행 후 관리자 홈에서 '회원가입' 메뉴를 클릭해서 회원가입 양식이 화면에 출력되는지 확인



[그림 10] '회원가입' 메뉴 실행화면

■ 컨트롤러 기능 구현

- 회원가입에 필요한 관리자 정보 입력 후 <create account> 버튼을 클릭하면 <form>의 action에 명시한 서버 주소로 관리자 정보가 전송됨
- 이를 서버에서 처리하기 위해 AdminMemberController에 createAccountConfirm() 메서드 추가하기
- createAccountConfirm()
  - ✓ @RequestMapping에 의해서 /createAccountConfirm 요청을 처리할 수 있음
  - ✓ 이때 클라이언트에서 서버로 데이터가 전송되는 방식은 post
  - ✓ 따라서 method = RequestMethod.POST를 반드시 명시해야 함

■ 컨트롤러 기능 구현

- createAccountConfirm

코드 9-5 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberController.java

```
01 @Controller
02 @RequestMapping("/admin/member")
03 public class AdminMemberController {
04     ...생략...
05     // 회원가입 확인
06     @RequestMapping(value = "/createAccountConfirm", method = RequestMethod.POST)
07     public String createAccountConfirm() {
08         System.out.println("[AdminMemberController] createAccountConfirm()");
09
10         return null;
11     }
12 }
```

@RequestMapping을 더 간략하게 코딩할 수는 없을까요?

가능합니다. @RequestMapping은 클라이언트의 요청(/createAccountConfirm)과 방식(POST)을 설정하고 이를 메서드(createAccountConfirm())에 연결mapping하는 역할을 합니다. 이를 보다 간략하게 하고자 @RequestMapping과 method 속성을 결합한 애너테이션으로 @PostMapping이 있습니다. @Post Mapping은 이름에서도 알 수 있듯이 post 방식의 클라이언트 요청을 메서드에 연결하는 역할을 합니다.

따라서 AdminMemberController.java의 createAccountConfirm()을 다음과 같이 간략하게 매핑해도 됩니다.

```
@RequestMapping(value = "/createAccountConfirm", method = RequestMethod.POST)
public String createAccountConfirm() {...}
```

```
@PostMapping("/createAccountConfirm")
public String createAccountConfirm() {...}
```

createAccountForm()은 get 방식의 요청으로 @PostMapping이 아닌 @GetMapping을 이용합니다. 앞으로는 @PostMapping과 @GetMapping을 사용합니다.

```
@RequestMapping(value = "/createAccountForm", method = RequestMethod.GET)
public String createAccountForm() {...}
```

```
@GetMapping("/createAccountForm")
public String createAccountForm() {...}
```

■ 컨트롤러 기능 구현

- createAccountConfirm()
  - ✓ 관리자가 회원가입 양식에 입력한 정보를 파라미터로 받아야 함
  - ✓ 이를 위해 VO 객체인 AdminMemberVo를 이용함

코드 9-6 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberVo.java

```
01 package com.office.library.admin.member;
02
03 public class AdminMemberVo {
04
05     int a_m_no;              // 관리자 번호
06     int a_m_approval;        // 최고 관리자 승인 여부
07     String a_m_id;           // 관리자 아이디
08     String a_m_pw;           // 관리자 비밀번호
09     String a_m_name;         // 관리자 이름
10     String a_m_gender;       // 관리자 성별 구분
11     String a_m_part;         // 관리자 근무 부서
12     String a_m_position;     // 관리자 업무
13     String a_m_mail;         // 관리자 메일
14     String a_m_phone;        // 관리자 연락처
15     String a_m_reg_date;     // 관리자 등록일
16     String a_m_mod_date;     // 관리자 수정일
17
```

```
18      public int getA_m_no() { return a_m_no; }
19      public void setA_m_no(int a_m_no) { this.a_m_no = a_m_no; }
20
21      public int getA_m_approval() { return a_m_approval; }
22      public void setA_m_approval(int a_m_approval) {
23          this.a_m_approval = a_m_approval;
24      }
25
26      public String getA_m_id() { return a_m_id; }
27      public void setA_m_id(String a_m_id) { this.a_m_id = a_m_id; }
28
29      public String getA_m_pw() { return a_m_pw; }
30      public void setA_m_pw(String a_m_pw) { this.a_m_pw = a_m_pw; }
31
32      public String getA_m_name() { return a_m_name; }
33      public void setA_m_name(String a_m_name) { this.a_m_name = a_m_name; }
34
35      public String getA_m_gender() { return a_m_gender; }
36      public void setA_m_gender(String a_m_gender) {
37          this.a_m_gender = a_m_gender;
38      }
39
40      public String getA_m_part() { return a_m_part; }
41      public void setA_m_part(String a_m_part) { this.a_m_part = a_m_part; }
42
43      public String getA_m_position() { return a_m_position; }
44      public void setA_m_position(String a_m_position) {
45          this.a_m_position = a_m_position;
46      }
```

```
47
48      public String getA_m_mail() { return a_m_mail; }
49      public void setA_m_mail(String a_m_mail) { this.a_m_mail = a_m_mail; }
50
51      public String getA_m_phone() { return a_m_phone; }
52      public void setA_m_phone(String a_m_phone) { this.a_m_phone = a_m_phone; }
53
54      public String getA_m_reg_date() { return a_m_reg_date; }
55      public void setA_m_reg_date(String a_m_reg_date) {
56          this.a_m_reg_date = a_m_reg_date;
57      }
58
59      public String getA_m_mod_date() { return a_m_mod_date; }
60      public void setA_m_mod_date(String a_m_mod_date) {
61          this.a_m_mod_date = a_m_mod_date;
62      }
63  }
```

■ 컨트롤러 기능 구현

- AdminMemberVo
  - ✓ 개인정보에 해당하는 멤버 필드와 이를 외부에서 사용할 수 있도록 setter/getter 메서드로 구성되어 있음
- AdminMemberController.java의 createAccountConfirm()을 수정함([코드 9-5] 7행)

```
public String createAccountConfirm(AdminMemberVo adminMemberVo) { ... }
```

  - ✓ 관리자가 입력한 정보를 AdminMemberVo에 담아 (setter 메서드 이용) createAccountConfirm()에 전달함
  - ✓ AdminMemberController까지 전달된 관리자 정보는 서비스 객체에 전달해야 함

■ 컨트롤러 기능 구현

- AdminMemberService.java 파일을 코딩하기

코드 9-7 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberService.java

```
01 package com.office.library.admin.member;
02
03 import org.springframework.stereotype.Service;
04
05 @Service
06 public class AdminMemberService {
07
08     public int createAccountConfirm(AdminMemberVo adminMemberVo) {
09         System.out.println("[AdminMemberService] createAccountConfirm()");
10
11         return 0;
12     }
13 }
```

■ 컨트롤러 기능 구현

- AdminMemberController에서 AdminMemberService 사용하기
  - ✓ AdminMemberService는 @Service에 의해 이미 스프링 컨테이너에 빈 객체로 생성되어 있으므로 AdminMemberController는 의존 객체 자동 주입 방법으로 AdminMemberSerive를 별도의 생성 과정 없이 사용할 수 있음
  - ✓ AdminMemberController에 @Autowired를 이용해서 AdminMemberSerive를 멤버 필드로 선언하는 코드

코드 9-8 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberController.java

```
01 @Controller
02 @RequestMapping("/admin/member")
03 public class AdminMemberController {
04
05     @Autowired
06     AdminMemberService adminMemberService;
07
08     ...생략...
09 }
```

■ 컨트롤러 기능 구현

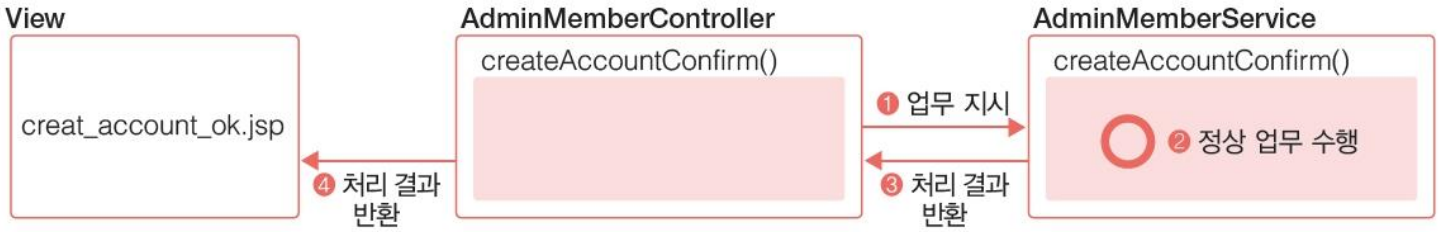
- createAccountConfirm()에서 AdminMemberService 사용하기
  - ✓ createAccountConfirm()에 다음과 같이 코드를 추가하기

코드 9-9 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberController.java

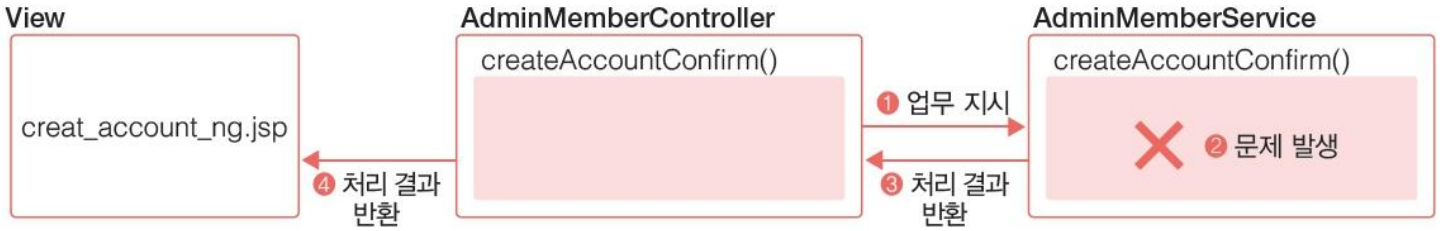
```
01 @Controller
02 @RequestMapping("/admin/member")
03 public class AdminMemberController {
04     ...생략...
05     // 회원가입 확인
06     @PostMapping("/createAccountConfirm")
07     public String createAccountConfirm(AdminMemberVo adminMemberVo) {
08         System.out.println("[AdminMemberController] createAccountConfirm()");
09
10         String nextPage = "admin/member/create_account_ok";
11
12         int result = adminMemberService.createAccountConfirm(adminMemberVo);
13
14         if (result <= 0)
15             nextPage = "admin/member/create_account_ng";
16
17         return nextPage;
18     }
19 }
```

■ 컨트롤러 기능 구현

- createAccountConfirm()
  - ✓ 내부적으로 adminMemberService의 createAccountConfirm()을 이용하고, adminMemberVo를 전달함
  - ✓ 반환 값 result
    : '관리자 회원가입' 업무가 정상적으로 처리됐다면 create_account_ok.jsp가 클라이언트한테 응답되고, 그렇지 않으면 create_account_ng.jsp가 응답됨



(a) 회원가입이 정상적으로 이루어지면 create_account_ok.jsp 응답



(b) 회원가입에 문제가 발생하면 create_account_ng.jsp 응답

[그림 11] 회원가입 처리에 따른 jsp 파일 호출

■ 서비스 기능 구현

- AdminMemberService의 createAccountConfirm()
  - ✓ AdminMemberService: 서비스 객체로 데이터베이스와 통신하기 위해 DAO를 이용함
  - ✓ 따라서 AdminMemberDao를 스프링 컨테이너에 빈 객체로 생성하고 AdminMemberService에서 @Autowired를 이용한 의존 객체 자동 주입을 함
  - ✓ 우선 AdminMemberDao.java 파일을 열어서 @Component를 추가하기

코드 9-10 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberDao.java

```
01 package com.office.library.admin.member;
02
03 import org.springframework.stereotype.Component;
04
05 @Component
06 public class AdminMemberDao {
07
08 }
```

■ 서비스 기능 구현

- AdminMemberService에 AdminMemberDao 빈 객체를 멤버 필드로 선언하고, @Autowired를 이용해서 의존 객체 자동 주입하기

코드 9-11 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberDao.java

```
01 @Service
02 public class AdminMemberService {
03
04     @Autowired
05     AdminMemberDao adminMemberDao;
06     ... 생략...
07
08 }
```

■ 서비스 기능 구현

- createAccountConfirm()의 업무는 '관리자 회원가입'으로 두 가지 작업을 해야 함

✓ 1. 사용자가 입력한 아이디가 기존에 다른 사람이 사용하고 있는 아이디와 중복되는지 확인함

- 중복 아이디라면 회원가입은 더 이상 진행할 수 없고, 서비스는 컨트롤러에 '회원가입 실패'를 알려줌

[그림 12] 중복 아이디라면 '실패' 메시지를 컨트롤러에 반환함

✓ 2. 중복 아이디가 아니라면 DAO를 통해서 데이터베이스에 관리자 정보를 추가해야 함

- DB에 모든 정보가 정상적으로 추가됐다면 서비스는 컨트롤러한테 '회원가입 성공'을 알려줌

[그림 13] 서비스는 회원정보가 정상, 비정상인지에 따라 회원가입 성공/실패를 알려줌

■ 서비스 기능 구현

- AdminMemberService.java에 두 가지 작업에 대한 코딩
  - ✓ createAccountConfirm()이 0 이하의 값(-1 또는 0)을 컨트롤러에게 반환하게 되면 관리자 회원가입은 실패이고, 1을 반환하게 되면 회원가입은 성공임
  - ✓ AdminMemberDao에 중복 아이디를 체크하고 회원정보를 추가하는 기능을 구현하기

코드 9-12 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberService.java

```
01 package com.office.library.admin.member;
02
03 import org.springframework.beans.factory.annotation.Autowired;
04 import org.springframework.stereotype.Service;
05
06 @Service
07 public class AdminMemberService {
08
09     final static public int ADMIN_ACCOUNT_ALREADY_EXIST = 0;
10     final static public int ADMIN_ACCOUNT_CREATE_SUCCESS = 1;
11     final static public int ADMIN_ACCOUNT_CREATE_FAIL = -1;
12
13     @Autowired
14     AdminMemberDao adminMemberDao;
15
```

```
16      public int createAccountConfirm(AdminMemberVo adminMemberVo) {
17          System.out.println("[AdminMemberService] createAccountConfirm()");
18
19          boolean isMember = adminMemberDao.isAdminMember(adminMemberVo.getA_m_id());
20
21          if (!isMember) {
22              int result = adminMemberDao.insertAdminAccount(adminMemberVo);
23
24              if (result > 0)
25                  return ADMIN_ACCOUNT_CREATE_SUCCESS;
26              else
27                  return ADMIN_ACCOUNT_CREATE_FAIL;
28          } else {
29              return ADMIN_ACCOUNT_ALREADY_EXIST;
30          }
31      }
32  }
```

Section 04

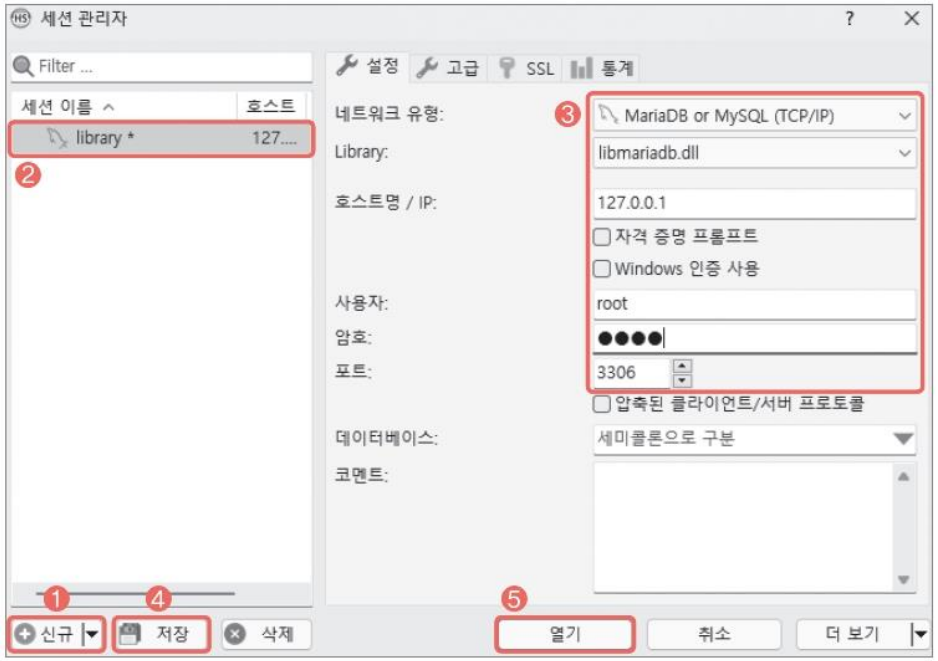
데이터베이스 만들고 연동하기

■ 데이터베이스

- AdminMemberDao의 역할은 데이터베이스(DB)와의 통신이므로 DB가 필요함
- 실습에서는 관계형 데이터베이스(relational database management system,RDBMS)인 MariaDB를 사용함
- MariaDB
  - ✓ MySQL과 동일한 소스 코드 기반으로 만들어졌으며, 표준 SQL을 따름

■ 하이디 SQL 설정하기

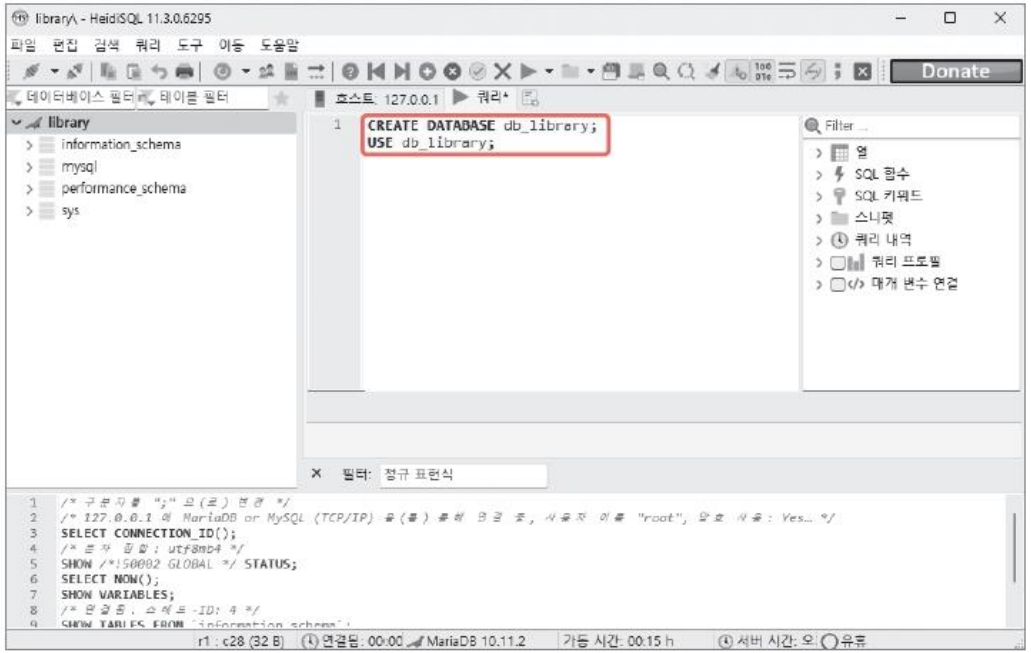
- 데이터베이스 관리툴인 하이디SQL(HeidiSQL)을 실행한 후 데이터베이스에 접속하기 위한 설정하기
- 1. 하이디SQL 실행 후 ❶ <신규> 버튼을 클릭. ❷ 이름은 library로 변경하고 ❸ 접속 정보(127.0.0.1, root, 3306)를 확인한 후 암호(1234)를 입력하기. 확인을 마쳤으면 ❹ <저장>과 ❺ <열기> 버튼을 순서대로 클릭하기.



■ 하이디 SQL 설정하기

- 2. library 데이터베이스 창이 열리면 상단의 쿼리를 클릭함
  - ✓ 프로젝트에서 사용할 데이터베이스 이름: 유_library
  - ✓ 디비 생성하고 선택하기

```
CREATE DATABASE db_library;  ———— db_library 데이터베이스 생성
USE db_library;              ———— db_library 데이터베이스 선택
```



■ 하이디 SQL 설정하기

- 3. 관리자 회원 정보를 관리하는 테이블을 명세서에 따라 tbl_admin_member 생성하기

[표 3]관리자 회원 테이블 명세서

관리자 회원 테이블(tbl_admin_member)					
컬럼명	데이터 타입	기본값	PK	NN	코멘트
a_m_no	INT		PK		번호(자동 증가)
a_m_approval	INT	0		NOT NULL	승인 여부(0: 승인 전, 1: 승인)
a_m_id	VARCHAR(20)			NOT NULL	아이디
a_m_pw	VARCHAR(100)			NOT NULL	비밀번호
a_m_name	VARCHAR(20)			NOT NULL	이름
a_m_gender	CHAR(1)			NOT NULL	성별 구분(M: 남성, W: 여성)
a_m_part	VARCHAR(20)			NOT NULL	근무부서
a_m_position	VARCHAR(20)			NOT NULL	직무
a_m_mail	VARCHAR(50)			NOT NULL	메일
a_m_phone	VARCHAR(20)			NOT NULL	연락처
a_m_reg_date	DATETIME				등록일
a_m_mod_date	DATETIME				수정일

```
CREATE TABLE tbl_admin_member(
    a_m_no          INT             AUTO_INCREMENT,
    a_m_approval    INT             NOT NULL DEFAULT 0,
    a_m_id          VARCHAR(20)     NOT NULL,
    a_m_pw          VARCHAR(100)    NOT NULL,
    a_m_name        VARCHAR(20)     NOT NULL,
    a_m_gender      CHAR(1)         NOT NULL,
    a_m_part        VARCHAR(20)     NOT NULL,
    a_m_position    VARCHAR(20)     NOT NULL,
    a_m_mail        VARCHAR(50)     NOT NULL,
    a_m_phone       VARCHAR(20)     NOT NULL,
    a_m_reg_date    DATETIME,
    a_m_mod_date    DATETIME,
    PRIMARY KEY(a_m_no)
    );
```

■ 하이디 SQL 설정하기

- 4. SELECT 쿼리로 tbl_admin_member 테이블이 정상적으로 생성됐는지 확인하기

```
SELECT * FROM tbl_admin_member;
```

tbl_admin_member (0r × 12c)

a_m_no	a_m_approval	a_m_id	a_m_pw	a_m_name	a_m_gender	a_m_part	a_m_position	a_m_mail	a_m_phone	a_m_reg_date	a_m_mod_date

JdbcTemplate

- 스프링에서는 JdbcTemplate 클래스를 제공함
- JdbcTemplate을 이용하면 'SQL 쿼리 작성 및 실행' 작업에 집중할 수 있는 장점이 있음
- JdbcTemplate을 사용하려면 메인 리포지터리에서 해당 모듈을 가져오고, 스프링 설정 파일을 이용해 JdbcTemplate 빈 객체를 스프링 컨테이너에 생성하는 작업이 필요함

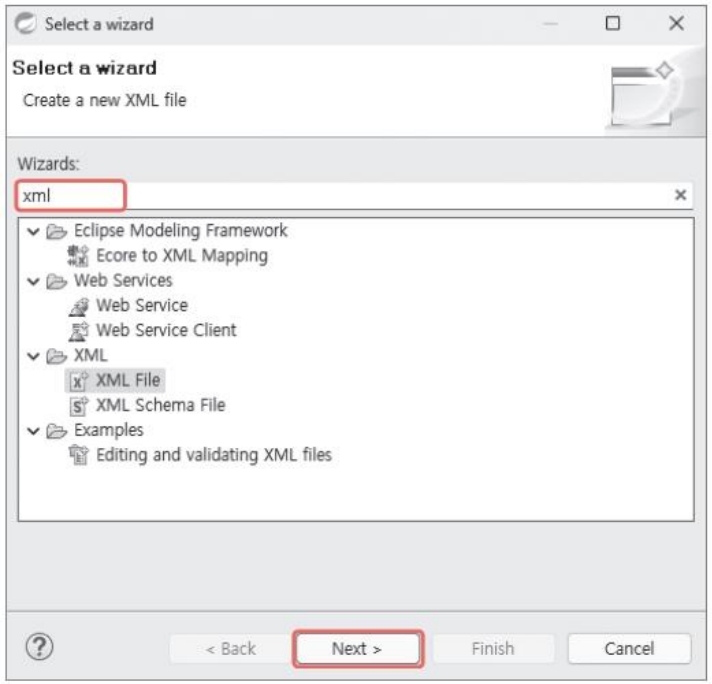
JdbcTemplate 빈 객체를 스프링 컨테이너에 생성

- 1. pom.xml에 spring-jdbc와 mariadb-java-client 모듈을 가져오기 위한 코드 추가

```
...생략...
    <!-- Test -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.7</version>
        <scope>test</scope>
    </dependency>

    <!-- JDBC -->
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${org.springframework-version}</version>
    </dependency>

    <!-- mariaDB -->
    <!-- https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client -->
    <dependency>
        <groupId>org.mariadb.jdbc</groupId>
        <artifactId>mariadb-java-client</artifactId>
        <version>3.1.2</version>
    </dependency>

</dependencies>
...생략...
```

JdbcTemplate 빈 객체를 스프링 컨테이너에 생성

- 2. jdbc-context.xml 파일을 새로 만들기 위해 [spring] 폴더에서 마우스 오른쪽 버튼을 클릭하여 [New]-[Other]를 클릭하기. [Select a wizard] 창에서 xml을 입력하여 XML File을 선택하고 <Next>를 클릭하기.
- 3. 파일명에 jdbc-context.xml을 입력하고 <Finish>를 클릭하기.



JdbcTemplate 빈 객체를 스프링 컨테이너에 생성

- 4. jdbc-context.xml 파일 코딩하기

코드 9-13 BookRentalPjt\src\main\webapp\WEB-INF\spring\jdbc-context.xml

```
01 <?xml version="1.0" encoding="UTF-8"
02 <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       ...생략...
       http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">
03
04 <!-- <context:property-placeholder
   location="/WEB-INF/spring/property/real.info.properties" /> -->
05
06 <!-- MariaDB connection 객체-->
07 <bean id="dataSource"
   class="org.springframework.jdbc.datasource.DriverManagerDataSource">
08     <property name="driverClassName" value="org.mariadb.jdbc.Driver" />
09     <property name="url" value="jdbc:mariadb://127.0.0.1:3306/db_library" />
10     <property name="username" value="root" />
11     <property name="password" value="1234" />
12 </bean>
13
14 <!-- JdbcTemplate 객체-->
15 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
16     <property name="dataSource" ref="dataSource" />
17 </bean>
18
19 <!-- TransactionManager 객체-->
20 <bean id="transactionManager"
   class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
21     <property name="dataSource" ref="dataSource" />
22 </bean>
23
24 </beans>
```

JdbcTemplate 빈 객체를 스프링 컨테이너에 생성

- 5. 프로젝트가 서버에서 실행될 때 jdbc-context.xml를 인식할 수 있도록 web.xml 파일에서 <context-param>에 jdbc-context.xml을 추가하기

```
<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>
```

```
<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/spring/root-context.xml
        /WEB-INF/spring/jdbc-context.xml
    </param-value>
</context-param>
```

AdminMemberDao에서 JdbcTemplate을 사용하기

- @Autowired를 이용해서 의존 객체 자동 주입하기
  - ✓ AdminMemberDao.java 수정하기

```
@Component
public class AdminMemberDao {

}
```

```
@Component
public class AdminMemberDao {

    @Autowired
    JdbcTemplate jdbcTemplate;

}
```

isAdminMember()

- 중복 아이디를 체크하는 메서드

코드 9-14 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberDao.java

```
01 @Component
02 public class AdminMemberDao {
03     @Autowired
04     JdbcTemplate jdbcTemplate;
05
06     public boolean isAdminMember(String a_m_id) {
07         System.out.println("[AdminMemberDao] isAdminMember()");
08
09         String sql =  "SELECT COUNT(*) FROM tbl_admin_member "
                       + "WHERE a_m_id = ?";
10
11         int result = jdbcTemplate.queryForObject(sql, Integer.class, a_m_id);
12
13         if (result > 0)
14             return true;
15         else
16             return false;
17         }
18 }
```

insertAdminAccount()

- isAdminMember()를 이용해서 중복 아이디 확인 결과가 true이면 tbl_admin_member 테이블에 관리자 정보를 추가하기 위한 메서드

코드 9-15 BookRentalPjt\src\main\java\com\office\library\admin\member\AdminMemberDao.java

```
01 @Component
02 public class AdminMemberDao {
03     ...생략...
04     public int insertAdminAccount(AdminMemberVo adminMemberVo) {
05         System.out.println("[AdminMemberDao] insertAdminAccount()");
06 
07         List<String> args = new ArrayList<String>();
08 
09         String sql =  "INSERT INTO tbl_admin_member(";
10             if (adminMemberVo.getA_m_id().equals("super admin")) {
11                 sql += "a_m_approval, ";
12                 args.add("1");
13             }
14 
15             sql += "a_m_id, ";
16             args.add(adminMemberVo.getA_m_id());
17 
18             sql += "a_m_pw, ";
19             args.add(adminMemberVo.getA_m_pw());
```

```
20
21              sql += "a_m_name, ";
22              args.add(adminMemberVo.getA_m_name());
23
24              sql += "a_m_gender, ";
25              args.add(adminMemberVo.getA_m_gender());
26
27              sql += "a_m_part, ";
28              args.add(adminMemberVo.getA_m_part());
29
30              sql += "a_m_position, ";
31              args.add(adminMemberVo.getA_m_position());
32
33              sql += "a_m_mail, ";
34              args.add(adminMemberVo.getA_m_mail());
35
36              sql += "a_m_phone, ";
37              args.add(adminMemberVo.getA_m_phone());
38
39              sql += "a_m_reg_date, a_m_mod_date) ";
40
41              if (adminMemberVo.getA_m_id().equals("super admin"))
42                  sql += "VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, NOW(), NOW())";
43              else
44                  sql += "VALUES(?, ?, ?, ?, ?, ?, ?, ?, NOW(), NOW())";
45
46          int result = -1;
47
```

```
48          try {
49
50              result = jdbcTemplate.update(sql, args.toArray());
51
52          } catch (Exception e) {
53              e.printStackTrace();
54          }
55          return result;
56      }
57  }
```

insertAdminAccount()

- 이 메서드는 보안상의 문제가 존재함
  - ✓ [예] 관리자가 비밀번호로 '1234'를 입력하면 'a_m_pw' 컬럼에 '1234'가 들어감
  - ✓ 비밀번호와 같이 중요한 정보는 DB에 암호화하여 추가하고 이를 다시 조회할 때 복호화해야 함

spring-security-core

- 데이터를 암호화하고 복호화하는 모듈
- spring-security-core 사용하기
  - ✓ pom.xml 파일에 모듈을 설정하고 [spring] 폴더에 security-context.xml 파일을 만들어 BCryptPasswordEncoder 빈 객체를 생성하는 코드 작성하기

[그림 14] 암호화와 복호화를 통한 비밀번호 관리

spring-security-core

- 1. pom.xml 파일을 열어 앞에서 추가한 mariadb-java-client 아래에 spring-security-core 모듈을 가져오기 위한 코드 추가하기

```
...생략...
    <!-- mariaDB -->
    <!-- https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client -->
    <dependency>
        <groupId>org.mariadb.jdbc</groupId>
        <artifactId>mariadb-java-client</artifactId>
        <version>3.1.2</version>
    </dependency>

    <!-- Spring security -->
    <!-- https://mvnrepository.com/artifact/org.springframework.security/spring-
security-core -->
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-core</artifactId>
        <version>${org.springframework-version}</version>
    </dependency>

</dependencies>
...생략...
```

spring-security-core

- 2. [spring] 폴더에 security-context.xml 파일을 새로 생성하고 코딩하기

코드 9-16 BookRentalPjt\src\main\webapp\WEB-INF\spring\security-context.xml

```
01 <?xml version="1.0" encoding="UTF-8"
02 <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       ...생략...
       http://www.springframework.org/schema/context/spring-context-4.3.xsd">
03
04 <bean id="bCryptPasswordEncoder"
   class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>
05
06 </beans>
```

spring-security-core

- 3. 프로젝트가 서버에서 실행될 때 security-context.xml을 인식할 수 있도록 web.xml 파일의 <context-param>에 security-context.xml을 추가하기

```
<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/spring/root-context.xml
        /WEB-INF/spring/jdbc-context.xml
        /WEB-INF/spring/security-context.xml
    </param-value>
</context-param>
```

spring-security-core

- 비밀번호를 암호화하기 위한 준비가 끝났다면 AdminMemberDao.java 수정하기
  - ✓ AdminMemberDao에 멤버 필드로 PasswordEncoder를 선언하고 의존 객체를 자동 주입하기

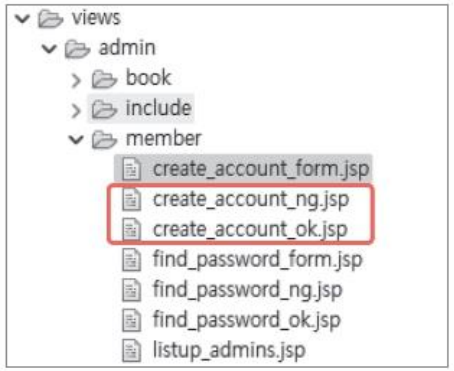
```
@Component
public class AdminMemberDao {

    @Autowired
    JdbcTemplate jdbcTemplate;

    @Autowired
    PasswordEncoder passwordEncoder;

    ...생략...
}
```

  - ✓ insertAdminAccount()에서 비밀번호를 암호화하는 부분([코드 9-15] 19행) 수정

```
sql += "a_m_pw, ";
args.add(passwordEncoder.encode(adminMemberVo.getA_m_pw()));
```

■ 관리자 회원가입 결과 화면

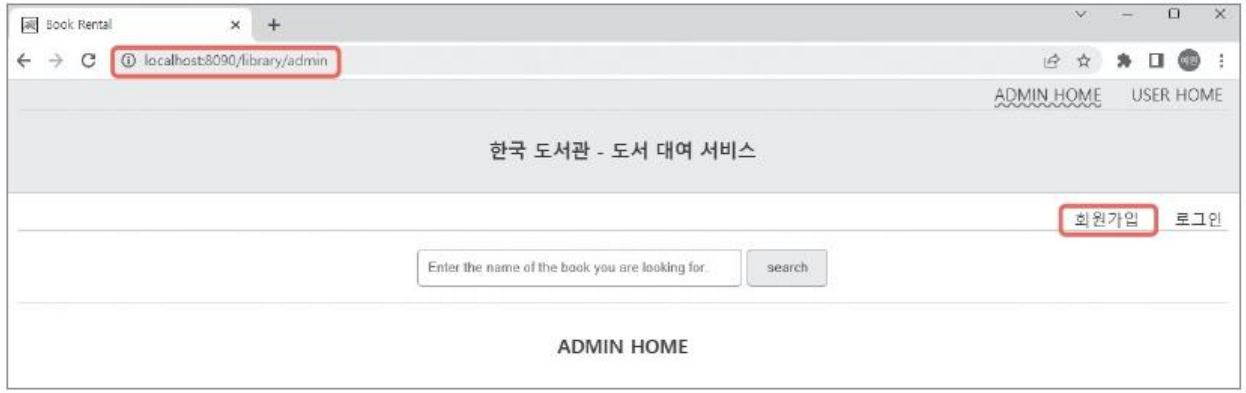
- 데이터베이스에 관리자 정보를 추가한 후 응답에 필요한 화면(뷰)을 생성
- 응답에 필요한 화면
  - ✓ [views]-[member] 폴더의 create_account_ok.jsp와 create_account_ng.jsp

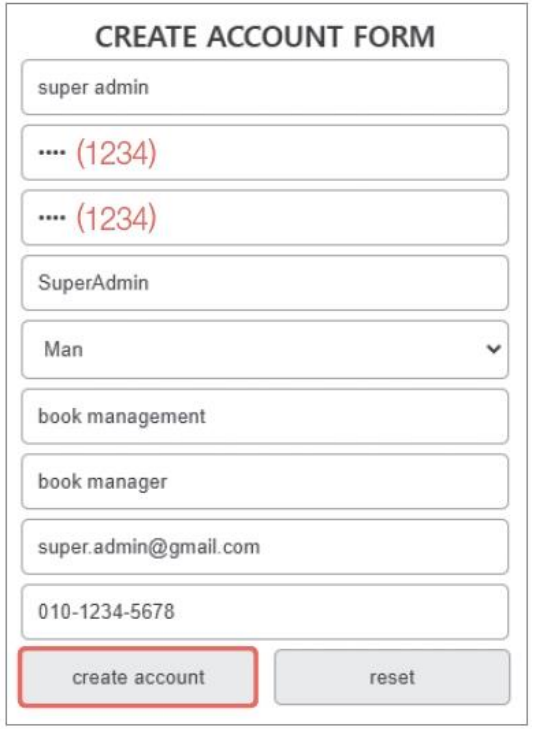


[그림 15] 관리자 회원가입 결과에 대한 응답 화면

■ 프로젝트를 실행하고 관리자 회원가입

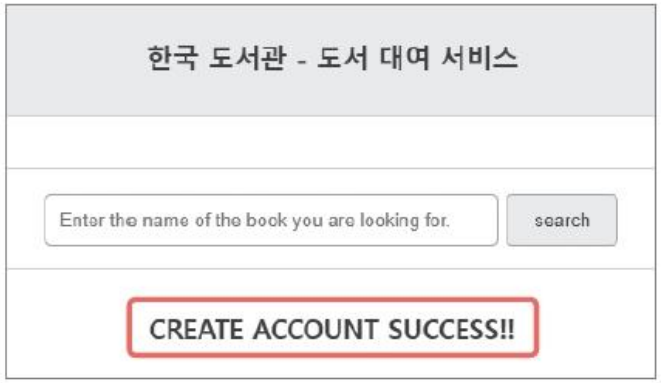
- 1. 프로젝트를 톰캣에서 실행하고 다음 URL에 접속해 '회원가입' 메뉴를 클릭
  - ✓ 접속 URL: http://localhost:8090/library/admin/
- 2. 회원가입 양식에 관리자 정보를 입력하기
  - ✓ 아이디: super admin (최고 관리자)
  - ✓ 비밀번호: 1234

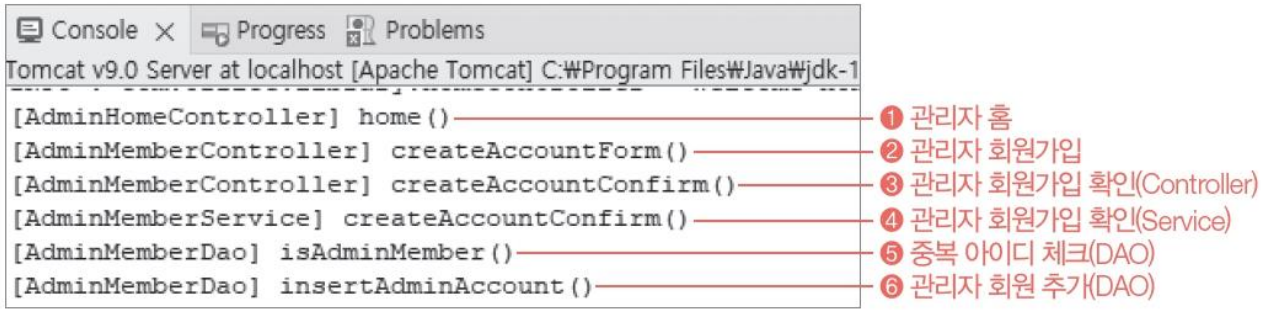




■ 프로젝트를 실행하고 관리자 회원가입을 진행

- 3. 회원가입 성공 화면 확인하기
- 4. 마지막으로 회원가입 진행 절차를 STS의 콘솔 창에 출력된 로그 확인하기





[그림 16] 출력된 로그

■ 프로젝트를 실행하고 관리자 회원가입을 진행

- 하이디SQL에서 tbl_admin_member 테이블 조회
  - ✓ DB에 최고 관리자(super admin) 회원 정보가 정상적으로 추가된 것을 확인하기
  - ✓ 특히 비밀번호 컬럼(a_m_pw)에 데이터가 암호된 것을 확인하기

```
SELECT * FROM tbl_admin_member;
```

tbl_admin_member (3r × 12c)

a_m_no	a_m_approval	a_m_id	a_m_pw	a_m_name	a_m_gender	a_m_part	a_m_position	a_m_mail	a_m_phone	a_m_reg_date	a_m_mod_date
1	1	super admin	$2a$10$EjbuQQin68rEppAO2sFwuOVvnh69/mIWE...	SuperAdmin	M	book management	book manager	super.admin@naver.com	+82-10-1234-5678	2023-05-02 15:57:37	2023-05-17 09:54:22