회원가입 구현하기

회원가입 구현하기

- **설명**

각 항목에 대한 유효성 검사는 클라이언트(JavaScript)와 백엔드(Spring MVC)에서 모두 수행 하는것이 좋음

클라이언트측에서 수행할 경우 사용자가 즉각적으로 유효성 검사의 결과를 확인하거나, 불필요한 서버 요청을 줄일 수 있음

서버측에서 수행할 경우 데이터 무결성을 보장함 (변조된 데이터를 검증할 수 있음)

그렇기 때문에 클라이언트와 서버측 모두 유효성 검사를 실시하는 것이 좋으나, 대부분의 회사는 자바스크립트로만 진행하는 곳이 많음

회원가입 구현하기

- 템플릿

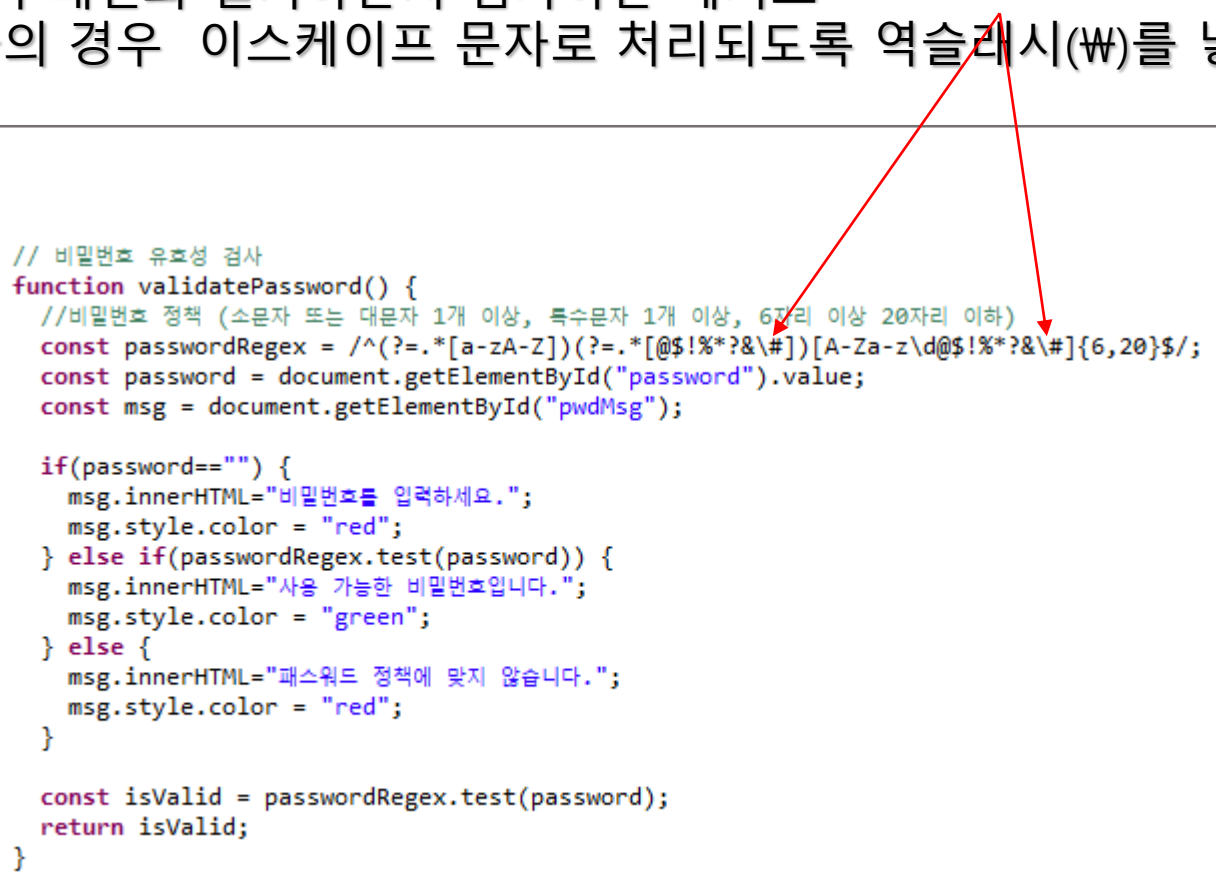
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!doctype html>
<html lang="ko" class="h-100">
<head>
<%@ include file="../common/head.jsp" %>
    <link rel="canonical" href="https://getbootstrap.kr/docs/5.2/examples/checkout-rtl/">
    <!-- Custom styles for this template -->
    <link href="/resources/css/auth/registerForm.css" rel="stylesheet">
  </head>

  <body class="text-center">
<main>
    <div class="py-5 text-center">
      <img class="d-block mx-auto mb-4" src="https://getbootstrap.kr/docs/5.2/assets/brand/bootstrap-logo.svg" alt="" width="72" height="57">
    <h2>Sign up</h2>
    </div>

    <div class="row g-3">
      <div class="">
        <form class="needs-validation" novalidate="">
          <div class="row g-3">
            <div class="col-12">
              <label for="email" class="form-label">이메일</label>
              <input type="email" class="form-control" id="email" placeholder="you@example.com">
              <div class="invalid-feedback">
                이메일
              </div>
              <span id="emailMsg"></span>
            </div>

            <div class="col-12">
              <label for="userName" class="form-label">이름</label>
              <input type="text" class="form-control" id="userName" placeholder="이름을 작성해주세요." required="">
              <div class="invalid-feedback">
                이름
              </div>
            </div>

            <div class="col-12">
              <label for="userName" class="form-label">비밀번호</label>
              <input type="password" class="form-control" id="password" onkeyup="validatePassword()" value="" required="">
              <div class="invalid-feedback">
                비밀번호
              </div>
              <span id="pwdMsg">대소문자 1개 이상, 특수문자 1개 이상, 6~20자리</span>
            </div>

            <div class="col-12">
              <label for="passwordChk" class="form-label">비밀번호 확인</label>
              <input type="password" class="form-control" id="passwordChk" onkeyup="checkPassword()" required="">
              <div class="invalid-feedback">
                비밀번호 확인
              </div>
              <span id="pwdChkMsg"></span>
            </div>

          <hr class="my-4">

          <button class="w-100 btn btn-primary btn-lg" type="submit">제출</button>
        </form>
      </div>
    </div>
  </main>
</body>
</html>
```

회원가입 구현하기
(클라이언트측 유효성 검사)

회원가입 구현하기 - 클라이언트

- **비밀번호 유효성 검사**

유효성 검사를 위한 정규식과 password의 값, 메시지를 넣을 id를 가져옴
* test : 정규 표현식 패턴과 일치하는지 검사하는 메서드
* 일부 특수문자들의 경우 이스케이프 문자로 처리되도록 역슬래시(₩)를 넣어주어야 함



```
// 비밀번호 유효성 검사
function validatePassword() {
  //비밀번호 정책 (소문자 또는 대문자 1개 이상, 특수문자 1개 이상, 6자리 이상 20자리 이하)
  const passwordRegex = /^(?=.*[a-zA-Z])(?=.*[@$!%*?&\#])[A-Za-z\d@$!%*?&\#]{6,20}$/;
  const password = document.getElementById("password").value;
  const msg = document.getElementById("pwdMsg");

  if(password=="") {
    msg.innerHTML="비밀번호를 입력하세요.";
    msg.style.color = "red";
  } else if(passwordRegex.test(password)) {
    msg.innerHTML="사용 가능한 비밀번호입니다.";
    msg.style.color = "green";
  } else {
    msg.innerHTML="패스워드 정책에 맞지 않습니다.";
    msg.style.color = "red";
  }

  const isValid = passwordRegex.test(password);
  return isValid;
}
```

회원가입 구현하기 - 클라이언트

- **비밀번호 유효성 검사**

^ : 문자열의 시작
(?=.*[a-zA-Z]) : 소문자 또는 대문자가 최소한 1개 이상 포함되어야 함
(?=.*[@$!%*?&₩#]) : 특수문자가 최소한 1개 이상 포함되어야 함. #은 이스케이프 처리
[A-Za-z₩d@$!%*&₩#]{6.20} : 대문자, 소문자, 숫자, 특수문자 중에서 6자리 이상 20자리 이하의 문자열이어야 함
$: 문자열의 끝

회원가입 구현하기 - 클라이언트

- **비밀번호 유효성 검사**

두 개의 비밀번호가 동일한지 확인

```
// 비밀번호 확인
function checkPassword() {
  const password = document.getElementById("password").value;
  const passwordChk = document.getElementById("passwordChk").value;
  const msg = document.getElementById("pwdChkMsg");

  if(password === passwordChk) {
    msg.innerHTML="패스워드가 동일합니다.";
    msg.style.color = "green";
  } else {
    msg.innerHTML="패스워드가 동일하지 않습니다.";
    msg.style.color = "red";
  }
}
```

회원가입 구현하기 - 클라이언트

- **문제점 확인**

  위와 같이 구현할 경우 문제가 되는 부분을 확인하기

회원가입 구현하기 - 클라이언트

- **문제점 해결**

비밀번호 확인 후 비밀번호 수정 시 텍스트가 변경되지 않기 때문에 아래와 같이 변경

```html
<div class="col-12">
  <label for="passwordChk" class="form-label">비밀번호 확인</label>
  <input type="password" class="form-control" id="passwordChk" onkeyup="validatePassword()" required="">
  <div class="invalid-feedback">
    비밀번호 확인
  </div>
  <span id="pwdChkMsg"></span>
</div>
```

```javascript
// 비밀번호 유효성 검사
function validatePassword() {
  //비밀번호 정책 (소문자 또는 대문자 1개 이상, 특수문자 1개 이상, 6자리 이상 20자리 이하)
  const passwordRegex = /^(?=.*[a-zA-Z])(?=.*[@$!%*?&\#])[A-Za-z\d@$!%*?&\#]{6,20}$/;
  const password = document.getElementById("password").value;
  const passwordChk = document.getElementById("passwordChk").value;
  const msg = document.getElementById("pwdMsg");
  const msgChk = document.getElementById("pwdChkMsg");

  if(passwordRegex.test(password)) {
    msg.innerHTML="사용 가능한 비밀번호입니다.";
    msg.style.color = "green";
  } else {
    msg.innerHTML="패스워드 정책에 맞지 않습니다.";
    msg.style.color = "red";
  }

  if(password === passwordChk) {
    msgChk.innerHTML="패스워드가 동일합니다.";
    msgChk.style.color = "green";
  } else {
    msgChk.innerHTML="패스워드가 동일하지 않습니다.";
    msgChk.style.color = "red";
  }

  const isValid = passwordRegex.test(password);
  return isValid;
}
```

회원가입 구현하기 - 클라이언트

- **registerForm.jsp**

ajax 통신을 위한 코드 작성

```
// 이메일 중복 확인 Ajax
function checkEmailAvailability() {
  const email = $("#email").val();
  const emailMsg = $("#emailMsg");
  const emailRegex = /^[A-Za-z0-9+_.-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$/;

  if(emailRegex.test(email)) {
    $.ajax({
      type: "POST",
      url: "/member/checkEmailAvailability.do",
      data: { email: email },
      success: function (response) {
        console.log(response);
        if (response == "N") {
          emailMsg.html("사용 가능한 이메일입니다.").css("color", "green");
        } else {
          emailMsg.html("이미 사용 중인 이메일입니다.").css("color", "red");
        }
      },
      error: function () {
        emailMsg.html("이메일 중복 확인 중 오류가 발생했습니다.").css("color", "red");
      }
    });
  } else {
    emailMsg.html("이메일 형식이 맞지 않습니다.").css("color", "red");
  }
}
```

회원가입 구현하기 - 클라이언트

- **kr.co.greenart.member.controller.MemberController**

@RequestParam을 사용하지 않는 이유
→ 현재 아래의 코드에서는 return을 "N" 또는 "Y"의 문자로 처리하려고 하지만,
RequestParam을 사용하면 해당하는 view를 찾아가기 때문에 에러가 발생함
→ ResponseBody는 리턴값을 HTTP 응답의 body로 직접 전송하여 뷰를 찾아가지 않고
요청을 보내온 클라이언트로 값을 직접 전달하게 됨

```
@PostMapping("/checkEmailAvailability.do")
@ResponseBody
public String checkEmailAvailability(String email) {
    System.out.println();
        // 이메일 중복 확인 로직을 수행하고 결과를 반환하는 부분을 구현합니다.
        int result = memberService.checkEmailAvailability(email);

        if(result>0) {
            return "Y";
        } else {
            return "N";
        }
 }
```

회원가입 구현하기 - 클라이언트

- kr.co.greenart.member.model.service.MemberService

```
public int checkEmailAvailability(String email) {
    return memberDao.checkEmailAvailability(sqlSession, email);
}
```

회원가입 구현하기 - 클라이언트

- kr.co.greenart.member.model.dao.MemberDao

```
public int checkEmailAvailability(SqlSessionTemplate sqlSession, String email) {
    return sqlSession.selectOne("memberMapper.checkEmailAvailability", email);
}
```

회원가입 구현하기 - 클라이언트

- member-mapper.xml

```
<select id="checkEmailAvailability" resultType="_int">
  SELECT COUNT(*) FROM member
  WHERE M_EMAIL = #{email}
</select>
```

회원가입 구현하기 (insert)

회원가입 구현하기 - insert

- **signup.jsp**

name 설정

```
<form class="needs-validation" action="/member/signup.do" method="post">
  <div class="row g-3">
    <div class="col-12">
      <label for="email" class="form-label">이메일</label>
      <input type="email" class="form-control" id="email" onkeyup="checkEmailAvailability()" name="memberEmail" placeholde
      <div class="invalid-feedback">
        이메일
      </div>
      <span id="emailMsg"></span>
    </div>

    <div class="col-12">
      <label for="userName" class="form-label">이름</label>
      <input type="text" class="form-control" id="userName" name="memberName" placeholder="이름을 작성해주세요." required="">
      <div class="invalid-feedback">
        이름
      </div>
    </div>

    <div class="col-12">
      <label for="userName" class="form-label">비밀번호</label>
      <input type="password" class="form-control" id="password" name="memberPassword" onkeyup="validatePassword()" value="
      <div class="invalid-feedback">
        비밀번호
      </div>
      <span id="pwdMsg">대소문자 1개 이상, 특수문자 1개 이상, 6~20자리</span>
    </div>

    <div class="col-12">
      <label for="passwordChk" class="form-label">비밀번호 확인</label>
      <input type="password" class="form-control" id="passwordChk" name="memberPasswordChk" onkeyup="validatePassword()" r
      <div class="invalid-feedback">
        비밀번호 확인
      </div>
      <span id="pwdChkMsg"></span>
    </div>
```

회원가입 구현하기 - insert

- **kr.co.greenart.member.model.dto.MemberDto**

jsp에서 작성한 name과 동일하게 필드명 설정

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Member {
    private int memberIdx;
    private String memberEmail;
    private String memberName;
    private String memberPassword;
    private String memberPasswordChk;
    private String memberIndate;
    private String memberRemoveDate;
}
```

회원가입 구현하기 - insert

- **kr.co.greenart.member.controller.MemberController**

전달받은 member 객체를 서비스에게 전달

```
@PostMapping("/signup.do")
public String signup(Member member) {
    // 회원가입
    int result = memberService.singupMember(member);

    if(result>0) {
        return "member/login";
    } else {
        return "common/errorPage";
    }
}
```

회원가입 구현하기 - insert

- kr.co.greenart.member.model.service.MemberService

```java
// 회원가입
int singupMember(Member member);
```

```java
public int singupMember(Member member) {
    return memberDao.singupMember(sqlSession, member);
}
```

회원가입 구현하기 - insert

- kr.co.greenart.member.model.dao.MemberDao

```
public int singupMember(SqlSessionTemplate sqlSession, Member member) {
    return sqlSession.insert("memberMapper.singupMember", member);
}
```

회원가입 구현하기 - insert

- member-mapper.xml

```
<insert id="singupMember" parameterType="member">
  INSERT INTO member(M_IDX,
                     M_EMAIL,
                     M_NAME,
                     M_PASSWORD,
                     M_INDATE,
                     M_REMOVE_DATE)
          VALUES (seq_m_idx.NEXTVAL,
                #{memberEmail},
                #{memberName},
                #{memberPassword},
                SYSDATE,
                NULL)
</insert>
```

회원가입 구현하기
(패스워드 암호화)

회원가입 구현하기 – 패스워드 암호화

- **pom.xml**

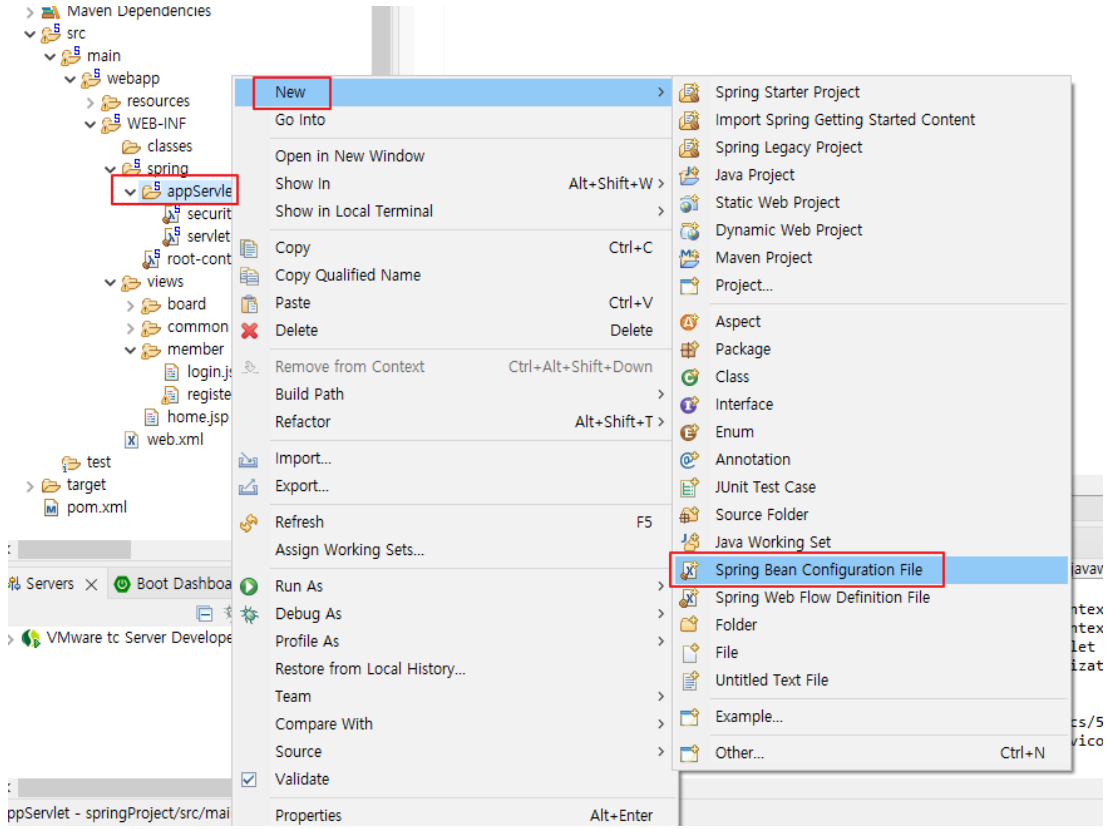
dependency 추가 후 프로젝트 우클릭 → Maven → Update Project
* 참고 : 너무 높은 버전을 사용하면 버전 이슈로 에러 발생할 수 있음

```
<!-- 패스워드 암호화 -->
<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId>spring-security-core</artifactId>
   <version>5.3.13.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.security/spring-security-web -->
<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId>spring-security-web</artifactId>
   <version>5.3.13.RELEASE</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.security/spring-security-config -->
<dependency>
   <groupId>org.springframework.security</groupId>
   <artifactId>spring-security-config</artifactId>
   <version>5.3.13.RELEASE</version>
</dependency>
```

회원가입 구현하기 – 패스워드 암호화

- **security-context.xml**

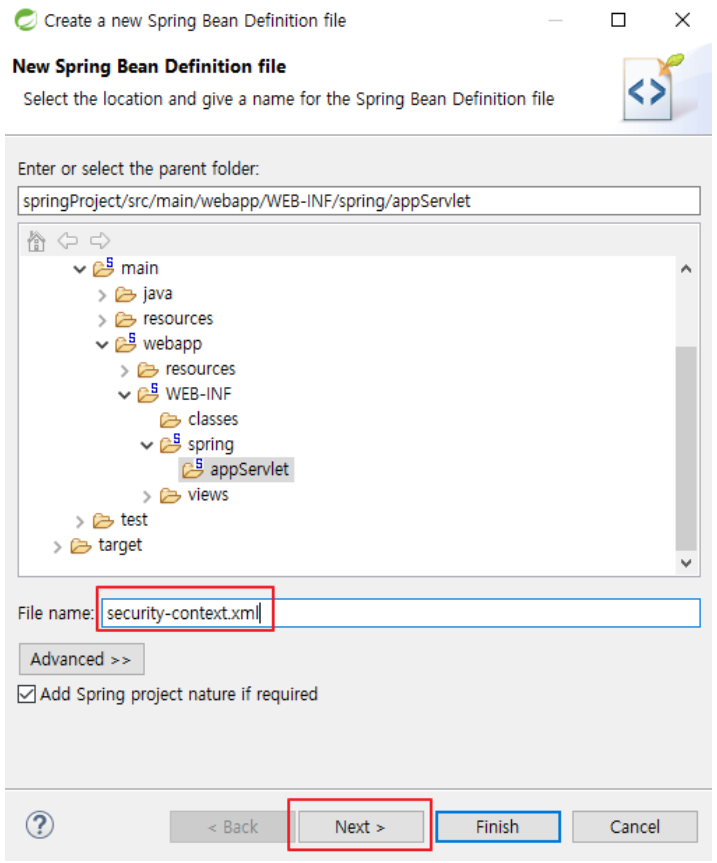
/WEB-INF/spring/appServlet 우클릭 → New → Spring Bean Configuration File 선택



회원가입 구현하기 – 패스워드 암호화

- **security-context.xml**

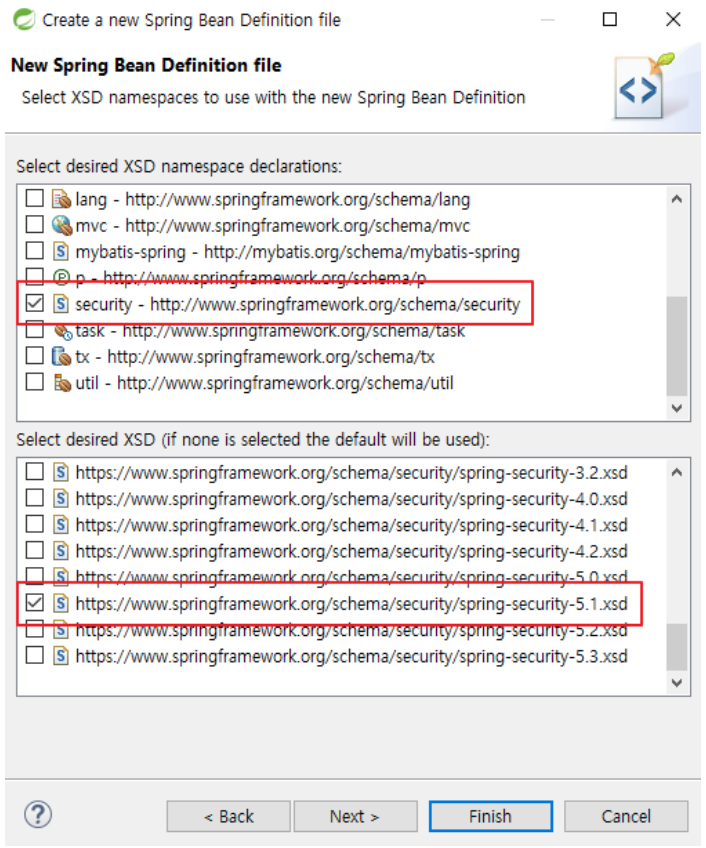
파일명 security-context.xml 작성 후 Next



회원가입 구현하기 – 패스워드 암호화

- **security-context.xml**

security 체크 → 아래 버전 맞는 xsd 체크 후 Finish



회원가입 구현하기 – 패스워드 암호화

- **security-context.xml**

아래 코드 복사+붙여넣기

<bean id="bcryptPasswordEncoder"
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:security="http://www.springframework.org/schema/security"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-5.3.xsd">

   <!-- bcryptPasswordEncoder -->
   <bean id="bcryptPasswordEncoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>

</beans>
```

회원가입 구현하기 – 패스워드 암호화

- **web.xml**

아래 코드 복사+붙여넣기

<param-value>/WEB-INF/spring/appServlet/security-context.xml</param-value>

```
<init-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/spring/appServlet/servlet-context.xml
    /WEB-INF/spring/appServlet/security-context.xml
  </param-value>
</init-param>
```

회원가입 구현하기 – 패스워드 암호화

- **kr.co.greenart.member.controller.MemberController**

Autowired 추가 후 signup.do에 암호화 관련 코드 작성

```
@Autowired
private BCryptPasswordEncoder bcryptPasswordEncoder;
```

```
@PostMapping("/signup.do")
public String signup(Member member) {
    // 중복 확인


    // 패스워드 암호화
    String password = bcryptPasswordEncoder.encode(member.getMemberPassword());
    member.setMemberPassword(password);

    // 회원가입
    int result = memberService.singupMember(member);

    if(result>0) {
        return "member/login";
    } else {
        return "common/errorPage";
    }
}
```

회원가입 구현하기 – 패스워드 암호화

- **암호화 확인**

서버 재시작 후 패스워드 암호화 확인

8	test4@test.com	홍길동4	$2a$10$eLKvu66R2rZETwH	2023-07-04 22:34:01.000
9	test5@test.com	홍길동5	$2a$10$O/aZ5OheUpXRep	2023-07-04 22:49:08.000

회원가입 구현하기
(서버측 유효성 검사)

회원가입 구현하기 – 서버측 유효성 검사

- 서버측 유효성 검사

```
@PostMapping("/signup.do")
public String signup(Member member) {
    // 중복 확인
    String password = member.getMemberPassword();
    String passwordChk = member.getMemberPasswordChk();
    String passwordRegex = "^(?=.*[a-zA-Z])(?=.*[@$!%*?&#])[A-Za-z\\d@$!%*?&#]{6,20}$";
    String email = member.getMemberEmail();

    // 이메일 검증
    String isEmailAvailable = checkEmailAvailability(email);

    // 패스워드 정책 검사, 패스워드=패스워드확인, 이메일 중복확인
    if(password.matches(passwordRegex) && password.equals(passwordChk) && isEmailAvailable.equals("N")) {
        // 패스워드 암호화
        String bcryptPassword = bcryptPasswordEncoder.encode(password);
        member.setMemberPassword(bcryptPassword);

        // 회원가입
        int result = memberService.singupMember(member);

        if(result>0) {
            return "member/login";
        } else {
            return "common/errorPage";
        }
    } else {
        return "common/errorPage";
    }
}
```

→ ajax로 요청하던 메서드에 email을 담고 결과값을 반환 받음

→ 변수 명 변경 (password → bcryptPassword)

회원가입 구현하기
(로그인)

회원가입 구현하기 – 로그인

- **member-mapper.xml**

```
<select id="loginMember" resultMap="memberResultSet">
  SELECT * FROM member
  WHERE M_EMAIL = #{memberEmail}
  AND M_PASSWORD = #{memberPassword}
  AND M_REMOVE_DATE IS NULL
</select>
```

→ 패스워드는 컨트롤러에서 비교하기 때문에 조건 제거

회원가입 구현하기 – 로그인

- **kr.co.greenart.member.controller.MemberController**

로그인 시 암호화된 패스워드를 비교하여 체크해야 함

view(login.jsp)에서 전달받은 패스워드

DB에서 불러온 암호화된 패스워드

```
@PostMapping("/login.do")
public String loginIndex(Member au, HttpSession session, Model model) {
    Member loginUser = memberService.loginMember(au);

    if(!Objects.isNull(loginUser) && bcryptPasswordEncoder.matches(au.getMemberPassword(), loginUser.getMemberPassword())) {
        session.setAttribute("memberIdx", loginUser.getMemberIdx());
        session.setAttribute("memberName", loginUser.getMemberName());
        session.setAttribute("msg", "로그인 되었습니다.");
        session.setAttribute("status", "success");
        return "redirect:/board/list.do"; // 페이지 리다이렉트
    } else {
        model.addAttribute("msg", "아이디 또는 비밀번호를 확인 해주세요.");
        model.addAttribute("status", "error");
        return "member/login";
    }
}
```