

JDBC + MyBatis

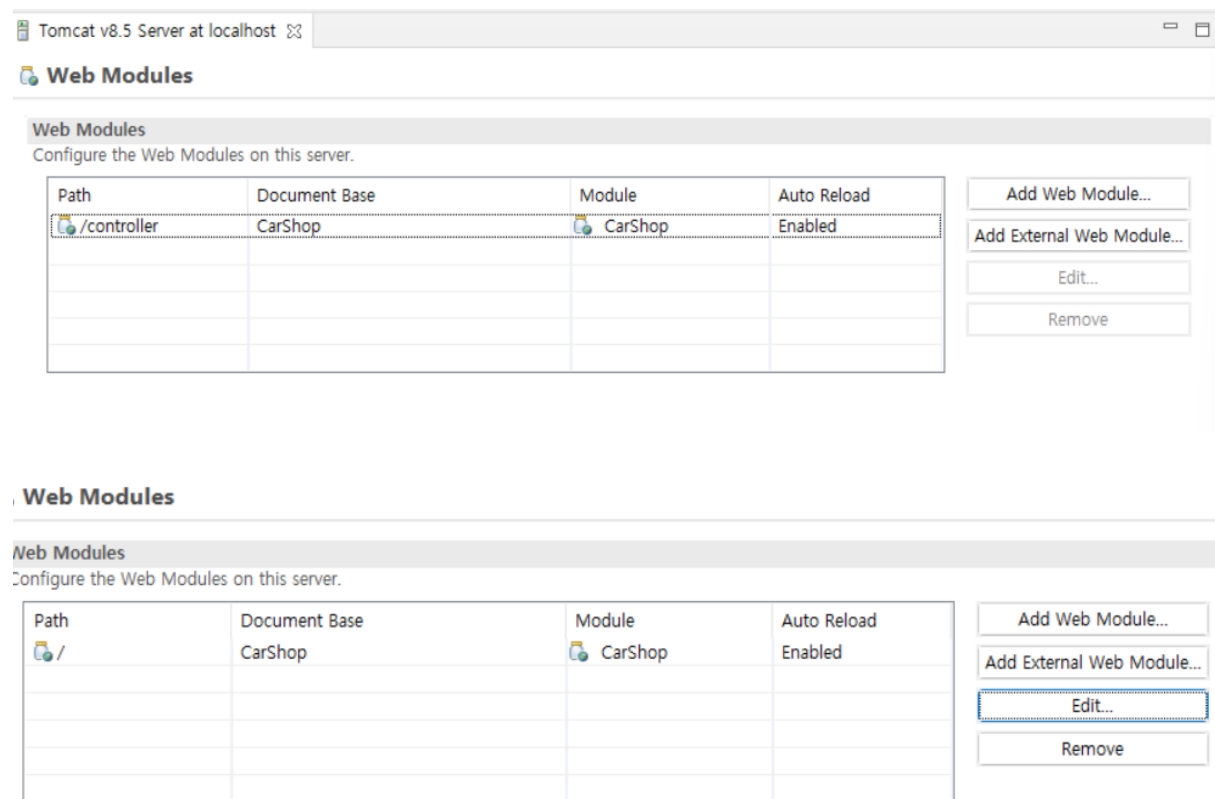
JDBC 만 적용 버전 JDBC + MyBatis 버전

[CarShop16.zip](#)

[CarShop17.zip](#)

JDBC 만 적용 버전으로 실습

새로운 프로젝트로 올리면 항상 경로 설정 2가



Tomcat v8.5 Server at localhost

Web Modules

Configure the Web Modules on this server.

Path	Document Base	Module	Auto Reload
/controller	CarShop	CarShop	Enabled

Add Web Module...
Add External Web Module...
Edit...
Remove

```
<!-- 업로드 패스 설정 -->
<beans:bean class="java.lang.String" id="uploadPath">
    <beans:constructor-arg
        value="C:\develop\spring\CarShop\src\main\webapp\resources" />
</beans:bean>
    <!--value="C:\develop\spring\.metadata\plugins\org.eclipse.wst.s
<!-- 일반 파일 업로드 경로 -->
```

```

<!-- 업로드 패스 설정 -->
<beans:bean class="java.lang.String" id="uploadPath">
    <beans:constructor-arg
        value="C:\dev\CarShop16\CarShop\src\main\webapp\resources" />
</beans:bean>
<!-- value="C:\dev\CarShop16\CarShop\src\main\webapp\resources" />

```

최종 동작 확인


홍차량 보기장바구니회원관리 ▾관리자 ▾

admin 님 로그인 하셨습니다. Log

장바구니

Shopping Cart

삭제하기 주문하기 »

제품	가격	수량	소계	비고
c0001-newsonata	5000	1	5000	삭제
총액			5000	

« 쇼핑 계속하기

DB 설정도 한번 확인하자... servlet-context.xml

```

77 <beans:bean id="dataSource"
78     class="org.springframework.jdbc.datasource.DriverManagerDataSource">
79     <beans:property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
80     <beans:property name="url" value="jdbc:mysql://localhost:3306/yonkeunsoo?serverTimezone=UTC"/>
81     <beans:property name="username" value="yonkeunsoo"/>
82     <beans:property name="password" value="guest0505"/>
83 </beans:bean>

```

MyBatis 기본설정

1.pom 의존성 라이브러리 6개 추가

```

<!-- <https://mvnrepository.com/artifact/org.mybatis/mybatis> -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.6</version>
</dependency>

<!-- <https://mvnrepository.com/artifact/org.mybatis/mybatis-spring> -->

```

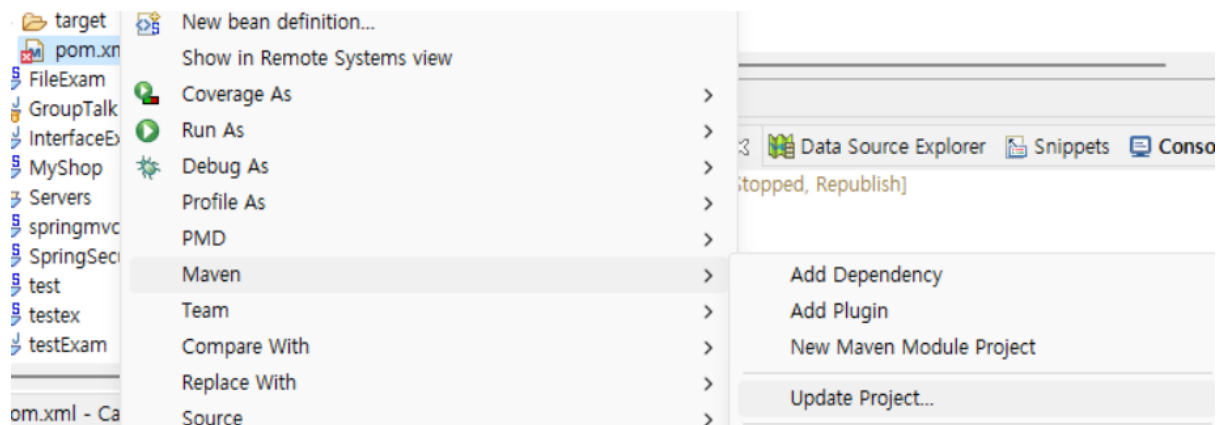
```

<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.4</version>
</dependency>
<!-- <https://mvnrepository.com/artifact/org.springframework/spring-jdbc> -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>5.3.19</version>
</dependency>
<!-- <https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2> -->
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-dbcp2</artifactId>
  <version>2.7.0</version>
</dependency>
<!-- <https://mvnrepository.com/artifact/org.bgee.log4jdbc-log4j2/log4jdbc-log4j2-jdbc4> -->
<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
  <version>1.16</version>
</dependency>

<!-- <https://mvnrepository.com/artifact/mysql/mysql-connector-java> -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.28</version>
</dependency>

```

pom.xml 을 수정하면 반드시 즉시 업데이트를 실시한다.



1. root-context.xml 3

```

<!-- mysql 연결 설정 -->
<bean id="dataSource"
  class="org.apache.commons.dbcp2.BasicDataSource"

```

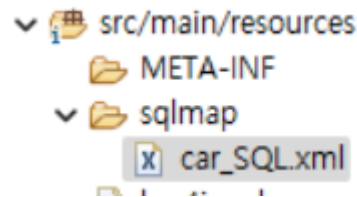
```

    destroy-method="close">
    <property name="driverClassName"
      value="com.mysql.cj.jdbc.Driver" />
    <property name="url"
      value="jdbc:mysql://localhost:3306/yoonkeunsoo?serverTimezone=UTC" />
    <property name="username" value="yoonkeunsoo" />
    <property name="password" value="guest0505" />
  </bean>

  <bean id="sqlSessionFactory"
    class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="mapperLocations"
      value="classpath:/sqlmap/**/*.xml" />
  </bean>
  <bean id="sqlSessionTemplate"
    class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg index="0" ref="sqlSessionFactory" />
  </bean>

```

1. Mapper 생성



```

<?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="car">

</mapper>

```

현재 예제는 JDBC 로 완벽 구현되어 있다.

일단 JDBC 를 비활성화 하자.

구현 삭제

```

@Repository
public class CarRepositoryImpl {
// public class CarRepositoryImpl implements CarRepository {

    private List<CarDTO> listOfCars = new ArrayList<CarDTO>();

// public CarRepositoryImpl() {
//

```

CRUD

```

<insert id="insert" parameterType="com.carshop.controller.CarDTO" useGeneratedKeys="true" keyProperty="cid">
<![CDATA[
    insert into car
    (cid, cname, cprice, ccate, cdesc, cfilename)
    values
    ({cid}, #{cname}, #{cprice}, #{ccate}, #{cdesc}, #{cfilename})
]]>

</insert>

```

@Repository 애너테이션을 잊지 말자!!!

```

@Repository
public class MyRepositoryImpl implements CarRepository {

```

```

@Autowired
sqlSessionTemplate sqlSessionTemplate;

@Override
public void setNewCar(CarDTO car) {
    this.sqlSessionTemplate.insert("car.insert", car);
}

```

CRUD

전체 목록

```

<select id="select_list"
    resultType="com.carshop.controller.CardTO">
    <![CDATA[
        select * from car
    ]]>
</select>

@Override
public List<CardTO> getAllCarList() {

    return this.sqlSessionTemplate.selectList("car.select_list");
}

```

상세 보기

```

<select id="select_detail" parameterType="String"
    resultType="com.carshop.controller.CardTO">
    <![CDATA[
        select * from car where cid = #{cid}
    ]]>

</select>

@Override
public CardTO getCarById(String cid) {

    return this.sqlSessionTemplate.selectOne("car.select_detail", cid);
}

```

CRUD

```

<update id="update1" parameterType="com.carshop.controller.CardTO">
    <![CDATA[
        update car set
            cname = #{cname},
            cprice = #{cprice},
            ccate = #{ccate},
            cdesc = #{cdesc},
            cfilename = #{cfilename}
        where cid = #{cid}
    ]]>
</update>

<update id="update2" parameterType="com.carshop.controller.CardTO">
    <![CDATA[
        update car set
            cname = #{cname},

```

```

        cprice = #{cprice},
        ccate = #{ccate},
        cdesc = #{cdesc}
        where cid = #{cid}
    ]]>
</update>

@Override
public void setUpdateCar(CardTO car) {
    if(car.getCfilename() != null) {
        this.sqlSessionTemplate.update("car.update1", car);
    } else if(car.getCfilename() == null) {
        this.sqlSessionTemplate.update("car.update2", car);
    }
}

```

CRUD

```

<delete id="delete" parameterType="String">
    <![CDATA[
        delete from car
        where cid = #{cid}
    ]]>
</delete>

```

인터페이스 전환 방법

클래스명 위에 @Primary 애너테이션 설정

```

@Primary
@Repository
public class MyRepositoryImpl implements CarRepository {

```

```

@Repository
public class CarRepositoryImpl implements CarRepository {

```

```

package com.carshop.controller;

import java.util.List;

import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Repository;

```

```

@Primary
@Repository
public class MyRepositoryImpl implements CarRepository {

    @Autowired
    SqlSessionTemplate sqlSessionTemplate;

    public List<CarDTO> getAllCarList() {

        return this.sqlSessionTemplate.selectList("car.select_list");
    }

    public List<CarDTO> getCarListByCategory(String category) {
        // TODO Auto-generated method stub
        return null;
    }

    public CarDTO getCarById(String cid) {

        return this.sqlSessionTemplate.selectOne("car.select_detail", cid);
    }

    public void setNewCar(CarDTO car) {
        this.sqlSessionTemplate.insert("car.insert", car);
    }

    public void removeCar(String cid) {
        this.sqlSessionTemplate.delete("car.delete", cid);
    }

    public void setUpdateCar(CarDTO car) {
        if(car.getCfilename() != null) {
            this.sqlSessionTemplate.update("car.update1", car);
        } else if(car.getCfilename() == null) {
            this.sqlSessionTemplate.update("car.update2", car);
        }
    }
}

<?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="car">

    <insert id="insert"
        parameterType="com.carshop.controller.CarDTO" useGeneratedKeys="true"
        keyProperty="cid">
        <![CDATA[
            insert into car
            (cid, cname, cprice, ccate, cdesc, cfilename)
            values
            (#{cid}, #{cname}, #{cprice}, #{ccate}, #{cdesc}, #{cfilename})
        ]]>

```



```

]]>

</insert>
<select id="select_list"
    resultType="com.carshop.controller.CardTO">
    <![CDATA[
        select * from car
    ]]>
</select>
<select id="select_detail" parameterType="String"
    resultType="com.carshop.controller.CardTO">
    <![CDATA[
        select * from car where cid = #{cid}
    ]]>

</select>

<update id="update1" parameterType="com.carshop.controller.CardTO">
    <![CDATA[
        update car set
            cname = #{cname},
            cprice = #{cprice},
            ccate = #{ccate},
            cdesc = #{cdesc},
            cfilename = #{cfilename}
        where cid = #{cid}
    ]]>
</update>

<update id="update2" parameterType="com.carshop.controller.CardTO">
    <![CDATA[
        update car set
            cname = #{cname},
            cprice = #{cprice},
            ccate = #{ccate},
            cdesc = #{cdesc}
        where cid = #{cid}
    ]]>
</update>
<delete id="delete" parameterType="String">
    <![CDATA[
        delete from car
        where cid = #{cid}
    ]]>
</delete>
</mapper>

```

Spring Security

호스트: 127.0.0.1 데이터베이스: yoonkeunsoo 테이블: [Untitled] 데이터 쿼리*

기본 옵션 인덱스 (1) 외래 키 (0) 제약 조건 확인 (0) 분할 CREATE 코드

추가	이름	유형/길이	알고리즘	코멘트
제거	PRIMARY KEY	PRIMARY		
초기화	username			Node

▲ 위로 ▼ 아래로

열: 추가 제거 ▲ 위로 ▼ 아래로

#	이름	데이터 유형	길이/설정	부호 없...	NULL 허용	0으로...	기본값
1	username	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음
2	password	VARCHAR	100	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음
3	enabled	TINYINT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0

```
CREATE TABLE `users` (
  `username` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `password` VARCHAR(100) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `enabled` TINYINT(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`username`) USING BTREE
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;

CREATE TABLE `authorities` (
  `username` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `authority` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
  UNIQUE INDEX `ix_auth_username` (`username`, `authority`) USING BTREE
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;

create unique index ix_auth_username on authorities (username,authority);

insert into users values ('austin', '1234', true);
insert into users values ('admin', '1234', true);
insert into authorities values ('admin', 'USER');
insert into authorities values ('admin', 'USER_MANAGER');
insert into authorities values ('austin', 'USER');

UPDATE users set password='$2a$10$lbdtX16tpVENpDSKwvs1h.wBPco2ZYEHsshWz5S44N.f8W/f1Hmja' WHERE username='austin';
UPDATE users set password='$2a$10$lbdtX16tpVENpDSKwvs1h.wBPco2ZYEHsshWz5S44N.f8W/f1Hmja' WHERE username='admin';
```

yooneunsoo.users: 2 행 (총) (대략적) >> 다음 << 모두 보기 < 정렬

username	password	enabled
admin	\$2a\$10\$ldTX16tpVENpDSKwws1h.wBPco2ZYEHsshWz5S44N.f8W/f1Hmja	1
austin	\$2a\$10\$ldTX16tpVENpDSKwws1h.wBPco2ZYEHsshWz5S44N.f8W/f1Hmja	1

기본 옵션 인덱스 (1) 외래 키 (0) 제약 조건 확인 (0) 분할 </> CREATE 코드 </> ALTER 코드

이름: authorities

코멘트:

열: + 추가 x 제거 ▲ 위로 ▼ 아래로

#	이름	데이터 유형	길이/설정	부호 없...	NULL 허...	0으로...	기본값	코멘트
1	username	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음	
2	authority	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	기본값 없음	

테이블이름과 컬럼 이름은 반드시 위와 동일하게 기본 설정하자.

특별히 수정을 안하고도 기본 값으로 위의 이름들이 스프링 시큐리티에 이미 정의 되어 있기 때문이다.

authorities 는 권한을 설정한다. db에 정의된 권한 정보를 시큐리티가 사용한다.

security-context.xml

```
<?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/security http://www.springframework.org/schema/security/spring-security.xsd
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- 각각의 intercept-url, form-login,logout 은 내부적으로 Filter를 만들어 사용한다. 그래서 web.xml에서 이 모든걸 엮어줄 FilterChain을 따로 설정해준다. -->
    <!-- web.xml에서 사용하는 FilterChain의 대한 설정부분이다. -->
    <security:http use-expressions="true">
        <security:intercept-url pattern="/cars/add/**" access="hasAuthority('USER_MANAGE R')"/>
        <security:intercept-url pattern="/**" access="permitAll"/>

        <security:form-login login-page="/login"
            default-target-url="/cars"
            authentication-failure-url="/loginfailed"

```

```

        username-parameter="username"
        password-parameter="password"/>
    </security:csrf />
    <security:logout logout-success-url="/logout"/>

</security:http>
<!-- form-login은 기본 로그인 폼 양식을 보여준다.logout은 로그아웃처리를.. -->

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

<!-- DB연동은 data-source만 지정해주면 된다, 테이블이름은 정확히. users 랑 authorities -->
<security:authentication-manager alias="authenticationManager">
    <security:authentication-provider>
        <security:jdbc-user-service data-source-ref="dataSource" />
        <security:password-encoder ref="passwordEncoder"/>
    </security:authentication-provider>
</security:authentication-manager>

<!-- 이것만 있으면 JDBC 코드 (Connection, Statement,ResultSet)로 DB연결 가능 -->
<bean id="dataSource"
class="org.apache.commons.dbcp2.BasicDataSource"
destroy-method="close">
    <property name="driverClassName"
        value="com.mysql.cj.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/yonkeunsoo?characterEncoding=utf8" />
    <property name="username" value="yonkeunsoo" />
    <property name="password" value="guest0505" />
    <property name="defaultAutoCommit" value="true" />
</bean>

</beans>

```

```

<?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="users">

    <insert id="insert"
        parameterType="com.carshop.users.User" useGeneratedKeys="true"
        keyProperty="username">
        <![CDATA[
            insert into users
            (username, password, enabled)
            values
            (#{username}, #{password}, 1)
        ]]>
    </insert>

```

```

    </mapper>

package com.carshop.users;

public interface UserRepository {

    void setNewUser(User user);
}

package com.carshop.users;

import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

@Repository
public class UserRepositoryImpl implements UserRepository {

    @Autowired
    SqlSessionTemplate sqlSessionTemplate;

    @Override
    public void setNewUser(User user) {
        this.sqlSessionTemplate.insert("users.insert", user);
    }
}

package com.carshop.users;

import org.springframework.stereotype.Service;

public interface UserService {

    void setNewUser(User user);
}

package com.carshop.users;

import org.springframework.beans.factory.annotation.Autowired;

public class UserServiceImpl implements UserService {

    @Autowired
    UserRepository userRepository;

    @Override
    public void setNewUser(User user) {

        userRepository.setNewUser(user);
    }
}

```

```

package com.carshop.users;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class UsersController {

    @Autowired
    UserService userService;

    @GetMapping("/join")
    public String joinForm(@ModelAttribute("NewUser") User user) {
        return "users/joinform";
    }

    @PostMapping("/join")
    public String submitForm(@ModelAttribute("NewUser") com.carshop.users.User user) {
        userService.setNewUser(user);
        return "redirect:/login";
    }
}

<form:form modelAttribute="NewUser"
    action="./join?${_csrf.parameterName}=${_csrf.token}"
    class="form-horizontal"
    method = "post">
    <fieldset>
    username : <form:input path="username" class="form-control"/>
    password : <form:input path="password" class="form-control"/>

    <input type="submit" class="btn btn-primary" value="등록"/>

    </fieldset>
</form:form>

```

호스트: 127.0.0.1	데이터베이스: yoonkeunsoo	테이블: users	데이터	쿼리*
yoonkeunsoo.users: 4 행 (총) (대략적)				
username	password	enabled		
admin	\$2a\$10\$lbTX16tpVENpDSK...	1		
austin	\$2a\$10\$lbTX16tpVENpDSK...	1		
test	test	1		
test21	test21	1		

주말 퀴즈

1. 암호가 암호화 처리가 되지 않아서 로그인이 안된다.
2. 암호를 암호화 처리를 해도 로그인이 안된다.