[한국ICT인재개발원] 스프링 프레임워크

2. DB 연동하기(MySQL)

강의자 소개 : 채종훈 강사

前) 광고데이터 분석 1년

前) IT강의 경력 2년 6개월

前) 머신러닝을 활용한 데이터 분석 프로젝트반 운영 1년

前) 리그오브 레전드 데이터 분석 등...

現) 국비반 강의 진행중

오늘 공부할 것들

1

MySQL Connector 설정과 테스트

MySQL Connector 설정하기
JUnit4를 활용한 테스트 코드 작성하기

2

커넥션풀 연동하기

Hikari cp 소개
Mysql 로그 콘솔창에 출력하기

3

MySQL 문법 체크하기

Oracle과 달라진 점 알아보기

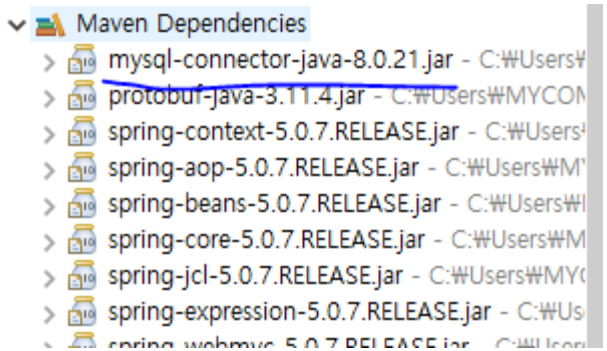
4

문제 풀어보기

ㄴ

MySQL 커넥터 설정


```
<!-- https://mvnrepository.com/artifact/mysql/mysc
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.21</version>
</dependency>
```



Mvnrepository에서 mysql을 검색해 설치된 버전과 맞는 커넥터 호출

1. **Pom.xml에 커넥터 jar파일 세팅하기.**
2. **Maven Dependencies에서 확인하기.**

테스트 코드 작성하기

JSP에서는 DB와 자바가 연결되었는지 확인하기 위해서는

반드시 DAO를 먼저 생성하고

서버를 돌려서 DB와 접속하는 로직을 이용해서

콘솔창에 뜨는 메시지를 보고 연결여부를 확인했었습니다.

그러나 테스트 코드를 작성하면, 서버 작동 없이도 확인할 수 있습니다.

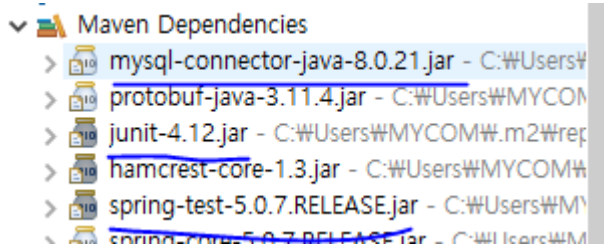
테스트 코드 사전 세팅

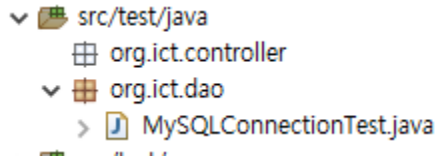

```
<!-- Test -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.0.7.RELEASE</version>
    <scope>test</scope>
</dependency>
```

1. Pom.xml에서 junit을 찾아 버전 4.7 -> 4.12로 변경하기
2. Mvnrepository에서 spring-test 검색해 스프링 버전과 동일하게 세팅

테스트 코드 생성





1. Junit과 spring-test가 제대로 들어왔는지 확인한 후
2. Test코드를 작성하는 src/test/java 하단에 테스트용 파일 생성

테스트 코드 생성


```
@Log4j
public class MySQLConnectionTest {
```

```
@Test
public void testConnection() thr

    Class.forName(DRIVER);

    try(Connection con = DriverM
```

1. 롬복의 Log4j 기능을 이용한 로깅을 위해 클래스명 위에 @Log4j 입력
2. 테스트 코드는 @Test가 붙은 메서드를 메인메서드처럼 간주합니다.

테스트 코드 생성


```java
@Log4j
public class MySQLConnectionTest {
    private final String DRIVER = "com.mysql.cj.jdbc.Driver";
    private final String URL =
            "jdbc:mysql://127.0.0.1:3306/mysql?useSSL=false&serverTimezone=UTC";
    private final String USER = "root";
    private final String PW = "1111";

    @Test
    public void testConnection() throws Exception {

        Class.forName(DRIVER);

        try(Connection con = DriverManager.getConnection(URL, USER, PW)) {

            log.info(con);
            log.info("mysql에 연결되었습니다");
```

1. 드라이버 설정
드라이버는 어떤 DB와 접속할지 컴퓨터에 안내해주는 용도로 지정합니다.

5버전 이하 => “com.mysql.jdbc.Driver”
6버전 이상 => “com.mysql.cj.jdbc.Driver”
로 입력하면 됩니다.

테스트 코드 생성


```java
@Log4j
public class MySQLConnectionTest {
    private final String DRIVER = "com.mysql.cj.jdbc.Driver";
    private final String URL =
            "jdbc:mysql://127.0.0.1:3306/mysql?useSSL=false&serverTimezone=UTC";
    private final String USER = "root";
    private final String PW = "1111";

    @Test
    public void testConnection() throws Exception {

        Class.forName(DRIVER);

        try(Connection con = DriverManager.getConnection(URL, USER, PW)) {

            log.info(con);
            log.info("mysql에 연결되었습니다");
```

2. DB주소 설정

DB주소는 설정에 따라 테이블명 자리에 들어갈 명칭이 달라집니다.

5버전 이하 => “jdbc:mysql://127.0.0.1:3306/테이블명”
6버전 이상
기본 => “jdbc:mysql://127.0.0.1:3306/테이블명?useSSL=false&serverTimezone=UTC”
기본 에러시 =>“jdbc:mysql://127.0.0.1:3306/mysql?useSSL=false&serverTimezone=UTC”

테스트 코드 생성


```java
@Log4j
public class MySQLConnectionTest {
    private final String DRIVER = "com.mysql.cj.jdbc.Driver";
    private final String URL =
            "jdbc:mysql://127.0.0.1:3306/mysql?useSSL=false&serverTimezone=UTC";
    private final String USER = "root";
    private final String PW = "1111";

    @Test
    public void testConnection() throws Exception {

        Class.forName(DRIVER);

        try(Connection con = DriverManager.getConnection(URL, USER, PW)) {

            log.info(con);
            log.info("mysql에 연결되었습니다");
```

3. DB내 계정 설정

계정명과 비밀번호를 저장합니다.

테스트 코드 생성


```java
@Test
public void testConnection() throws Exception {

    Class.forName(DRIVER);

    try(Connection con = DriverManager.getConnection(URL, USER, PW)) {

        log.info(con);
        log.info("mysql에 연결되었습니다");
    } catch(Exception e) {
        e.printStackTrace();
    }
}
```

4. 테스트 코드 작성

- @Test가 붙은 메서드를 테스트 코드 실행시 실제로 실행합니다.
- DBMS를 설정하는 class.forName(); 은 기본적으로 에러처리를 위해 throws Exception이 붙어야 합니다.
- 자바 1.7이후로 JDBC 코드는 반드시 try~catch구문을 사용해야합니다.

테스트 코드 생성


```
INFO : org.ict.dao.MySQLConnectionTest - com.mysql.cj.jdbc.ConnectionImpl@54bff557
INFO : org.ict.dao.MySQLConnectionTest - mysql에 연결되었습니다
```

```
java.sql.SQLSyntaxErrorException: Unknown database 'ictboard'
        at com.mysql.cj.jdbc.exceptions.SQLError.createSQLExce
        at com.mysql.cj.jdbc.exceptions.SQLError.createSQLExce
        at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.t
        at com.mysql.cj.jdbc.ConnectionImpl.createNewIO(Connec
        at com.mysql.cj.jdbc.ConnectionImpl.<init>(Connection]
        at com.mysql.cj.jdbc.ConnectionImpl.getInstance(Connec
        at com.mysql.cj.jdbc.NonRegisteringDriver.connect(NonF
        at java.sql/java.sql.DriverManager.getConnection(Drive
```

6. 정상 연결시 위와같이 커넥터 주소가 나오며 비정상 연결시 콘솔창에 익셉션이 발생하므로 참고하세요.

커넥션 풀 도입이 필요한 이유

커넥션 풀은 DAO가 하나인경우 접속자가 여럿 몰렸을 때

하나의 DAO가 모든 요청을 처리할 수 없어

지연이 발생하거나 서버가 터지는 것을 방지하기 위해

DAO를 요청량에 따라 가변적으로 생성하여

많은 요청도 여러 개의 DAO로 대응할 수 있도록 도와줍니다.

테스트 코드 생성

Hikari-cp는 spring-boot에서 현재 표준으로 채택된 커넥션 풀로

스프링 개발자들이 가장 많이 사용하는 커넥션 풀입니다.

설치를 위해 mvnrepository에서 hikaricp를 검색합니다.

테스트 코드 생성

Hikari-cp는 spring-boot에서 현재 표준으로 채택된 커넥션 풀로

스프링 개발자들이 가장 많이 사용하는 커넥션 풀입니다.

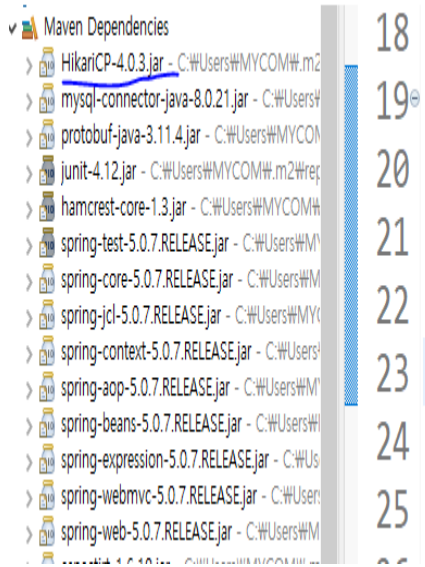
설치를 위해 mvnrepository에서 hikaricp를 검색합니다.

테스트 코드 생성

Artifacts

Java 8 thru 11 maven artifact:

```
<dependency>
    <groupId>com.zaxxer</groupId>
    <artifactId>HikariCP</artifactId>
    <version>4.0.3</version>
</dependency>
```



https://github.com/brettwooldridge/HikariCP
제작자 공식 깃허브에 4.0.3버전을 사용해도 무방하다고 하니 4.0.3버전 세팅을 합니다.

테스트 코드 생성


```
<!-- Root Context: defines shared resources visible to all other web components -
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
    <property name="driverClassName"
        value="com.mysql.cj.jdbc.Driver"></property>
    <property name="jdbcUrl"
        value="jdbc:mysql://localhost:3306/mysql?serverTimezone=UTC"></property>
    <property name="username" value="root"></property>
    <property name="password" value="1111"></property>
</bean>

<bean id="datasource"
        class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
    <constructor-arg ref="hikariConfig"></constructor-arg>
</bean>
```

root-context.xml 내부에 bean 설정을 합니다.
hikariCP에 접속정보를 입력한 다음
커넥션풀을 담당하는 DataSource 객체에 집어넣어줍니다.

테스트 코드 생성


```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
@Log4j
public class MySQLConnectionPoolTest {

    @Autowired
    private DataSource dataSource;

    @Test
    public void testConnection() {
        try(Connection con = dataSource.getConnection()){
            log.info(con);
            log.info("hikariCP connected");
        }catch(Exception e) {
            e.printStackTrace();
            }
        }
}
```

테스트용 코드로 MySQLConnectionPoolTest를 생성합니다.

커넥션 풀 설정이 된 코드는 DataSource 객체가 connection을 호출합니다.

테스트 코드 생성


```
org.springframework.test.context.support.DefaultTestContextBootstrapper -
org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XM
org.springframework.context.support.GenericApplicationContext - Refreshing
org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostPr
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
org.ict.dao.MySQLConnectionPoolTest - HikariProxyConnection@1727424614 wra
org.ict.dao.MySQLConnectionPoolTest - hikariCP connected
org.springframework.context.support.GenericApplicationContext - Closing or
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

연결이 정확하게 되었다면 위와 같이 HikariPool을 활용한 접속 로그가 나오며, 작업에 따라 HikariPool의 번호가 1번만 나오거나 1~n번까지 나오는 등 시스템 부하에 따른 조절을 합니다.