

# Springboot를 활용한 api서버 만들기

2019.03.21 | 석사과정 이요셉

# Environments

개발도구: VSCODE

DBMS: Mariadb

# 1. Springboot initializer

springboot는 IntelliJ와 같은 IDE나 vscode의 여러 extension에서 프로젝트를 생성할 수 있다.

하지만 이들의 동작 방식은 사실 모두 spring에서 제공하는 initializer를 사용하고 있다.

<https://start.spring.io/>

위 링크에 접속해서 attribute를 입력하고 간단하게 생성된 Springboot 프로젝트를 다운로드 받는다.  
Gradle 프로젝트를 기준으로 한다.

The screenshot shows the Spring Initializr web application interface. At the top, it says "Spring Initializr Bootstrap your application" with GitHub and Twitter links. The interface is divided into two main sections: "Project Metadata" and "Dependencies".

**Project Metadata:**

- Project:** Maven Project (selected) / Gradle Project
- Language:** Java (selected) / Kotlin / Groovy
- Spring Boot:** 2.2.0 M1 / 2.2.0 (SNAPSHOT) / 2.1.4 (SNAPSHOT) / 2.1.3 (selected) / 1.5.19
- Project Metadata:**
  - Group: aii.pilot
  - Artifact: qrmktree
  - Name: qrmktree
  - Description: Demo project for Spring Boot
  - Package Name: aii.pilot.qrmktree
  - Packaging:

A green button "Generate Project" with a download icon is visible.

**Dependencies:**

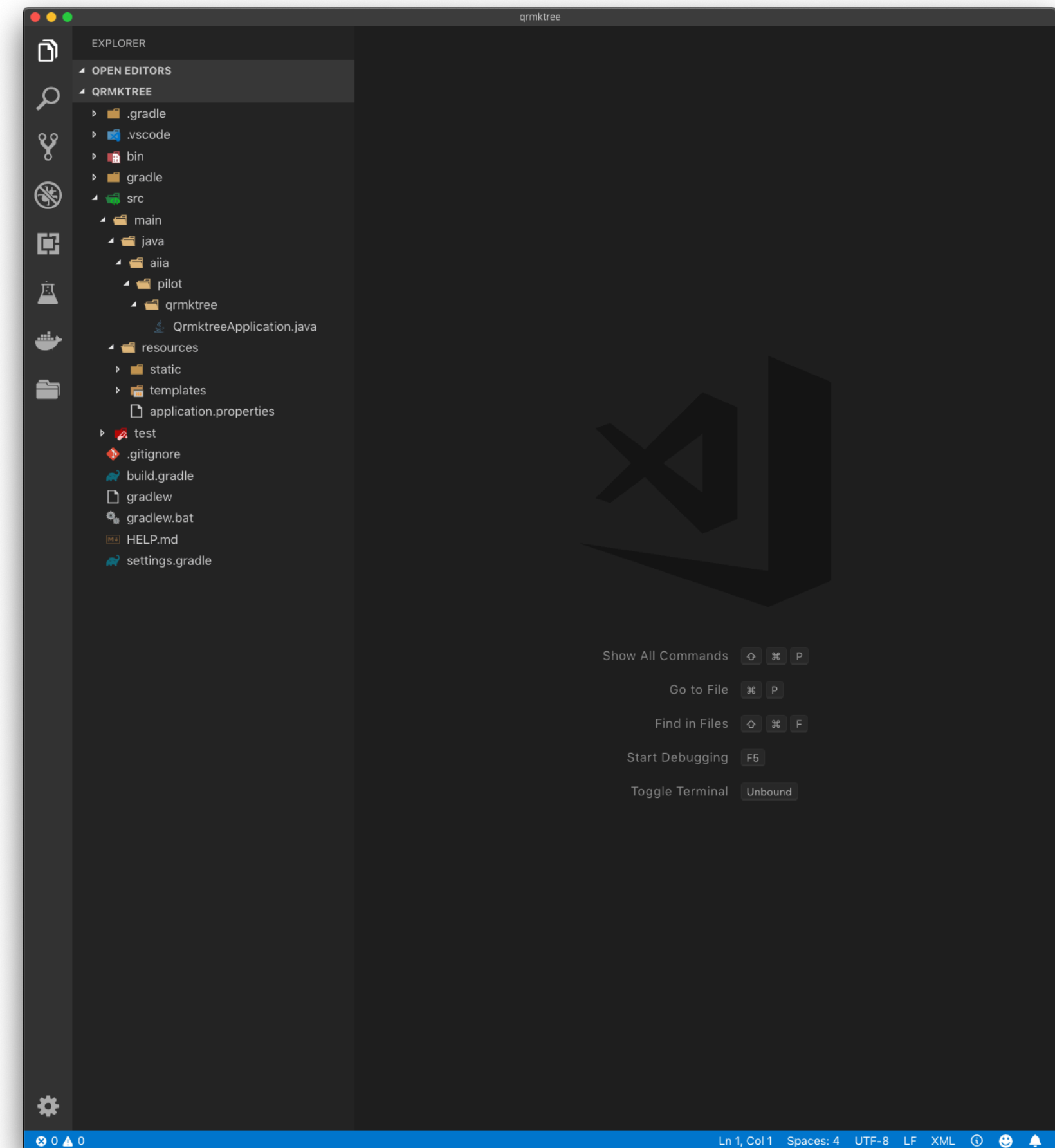
- Search dependencies to add:** Web, Security, JPA, Actuator, Devtools...
- Dependencies selected:**
  - Web [Web]:** Servlet web application with Spring MVC and Tomcat
  - DevTools [Core]:** Spring Boot Development Tools
  - MyBatis [SQL]:** Persistence support using MyBatis

Another green button "Generate Project" with a download icon is visible at the bottom.

## 2. Load on vscode

다운로드 받은 압축 형태의 프로젝트를 해제하고, 해당 폴더를 vscode에 로드한다.

오른쪽과 같은 프로젝트 디렉토리가 초기 상태이다.



# 3. Add dependencies

이번 프로젝트에 필요할 dependency 들을 gradle을 통해 관리될 수 있도록 한다. 아래 파일을 수정한다.

## build.gradle

### 1. jdbc 사용

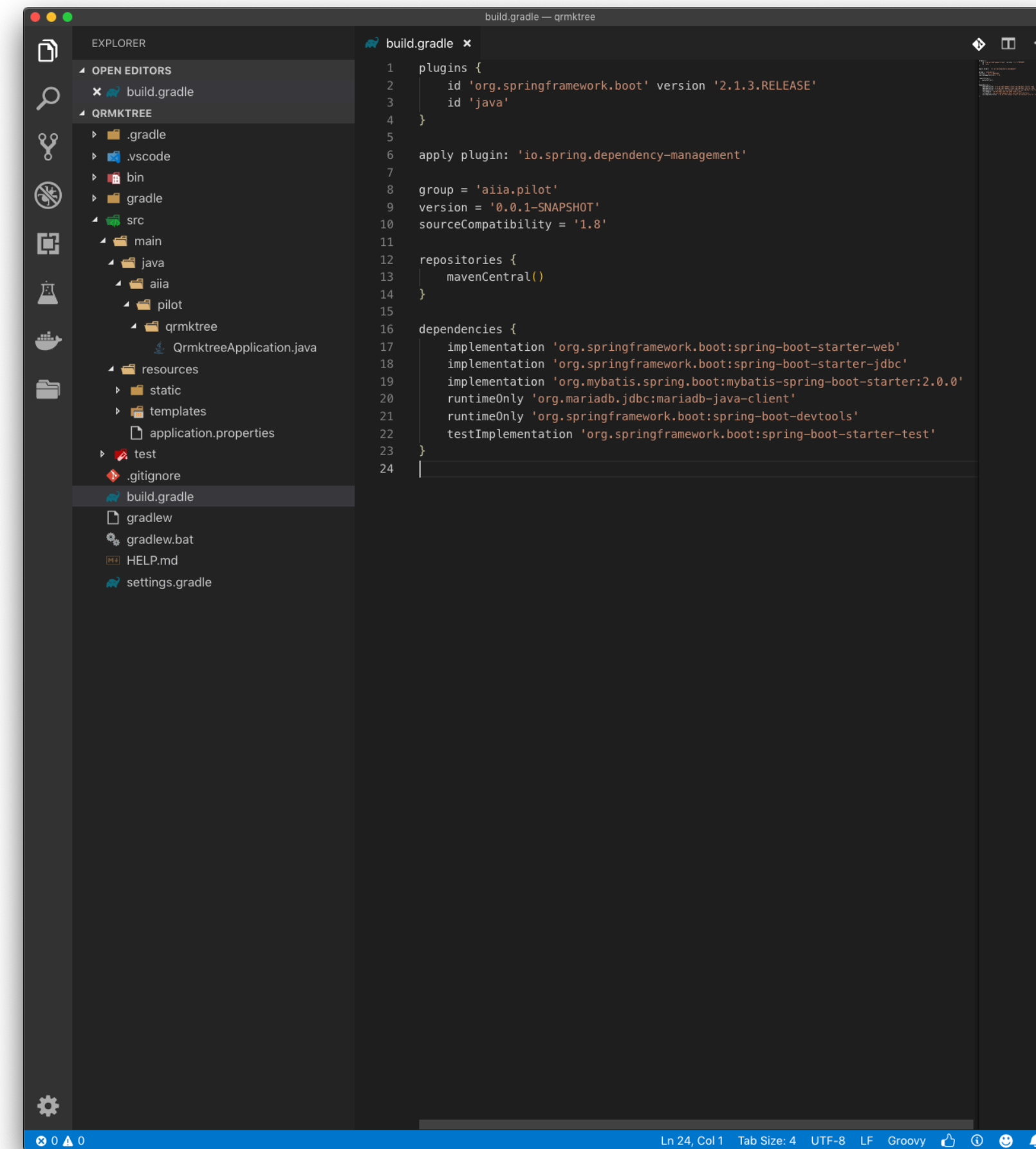
### 2. Mybatis 사용

- xml파일로 query문 관리

### 3. devtools사용

- 개발 효율성: 수정시 자동으로 서버에 반영함

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.springframework.boot:spring-boot-starter-jdbc'  
    implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.0.0'  
    runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'  
    runtimeOnly 'org.springframework.boot:spring-boot-devtools'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
```



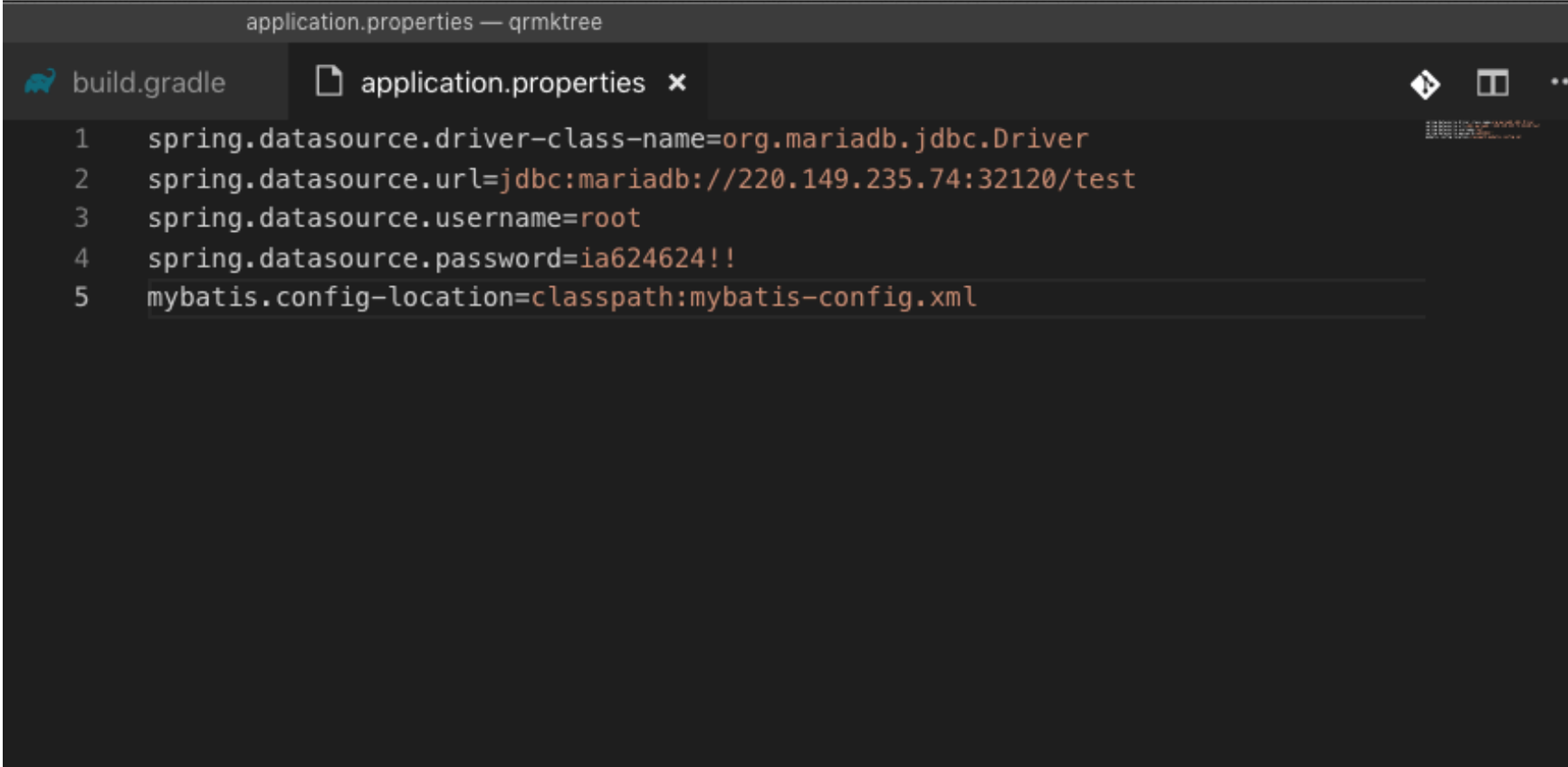
# 4. Spring Settings

Springboot의 가장 큰 특징중 하나는 Spring MVC의 복잡한 xml설정을 간소화하는 동시에 Spring의 기존 기능들을 사용할 수 있다는 것이다. 아래 파일을 수정하여 설정을 완료한다.

resources/application.properties

1. Datasource: mariadb와의 연결관계를 정의

2. Mybatis.config: mybatis 설정파일 위치를 정의



```
application.properties — qrmktree
build.gradle application.properties x
1 spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
2 spring.datasource.url=jdbc:mariadb://220.149.235.74:32120/test
3 spring.datasource.username=root
4 spring.datasource.password=ia624624!!
5 mybatis.config-location=classpath:mybatis-config.xml
```

# 5. Mybatis configuration

Mybatis 사용을 위해 아래 파일을 resources/에 생성한다.

Mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE configuration

    PUBLIC "-//mybatis.org//DTD Config 3.0//
    EN"

    "http://mybatis.org/dtd/mybatis-3-
    config.dtd">

<configuration>

    <mappers>

        <!-- <mapper resource="mapper/*.xml"/> -->

    </mappers>

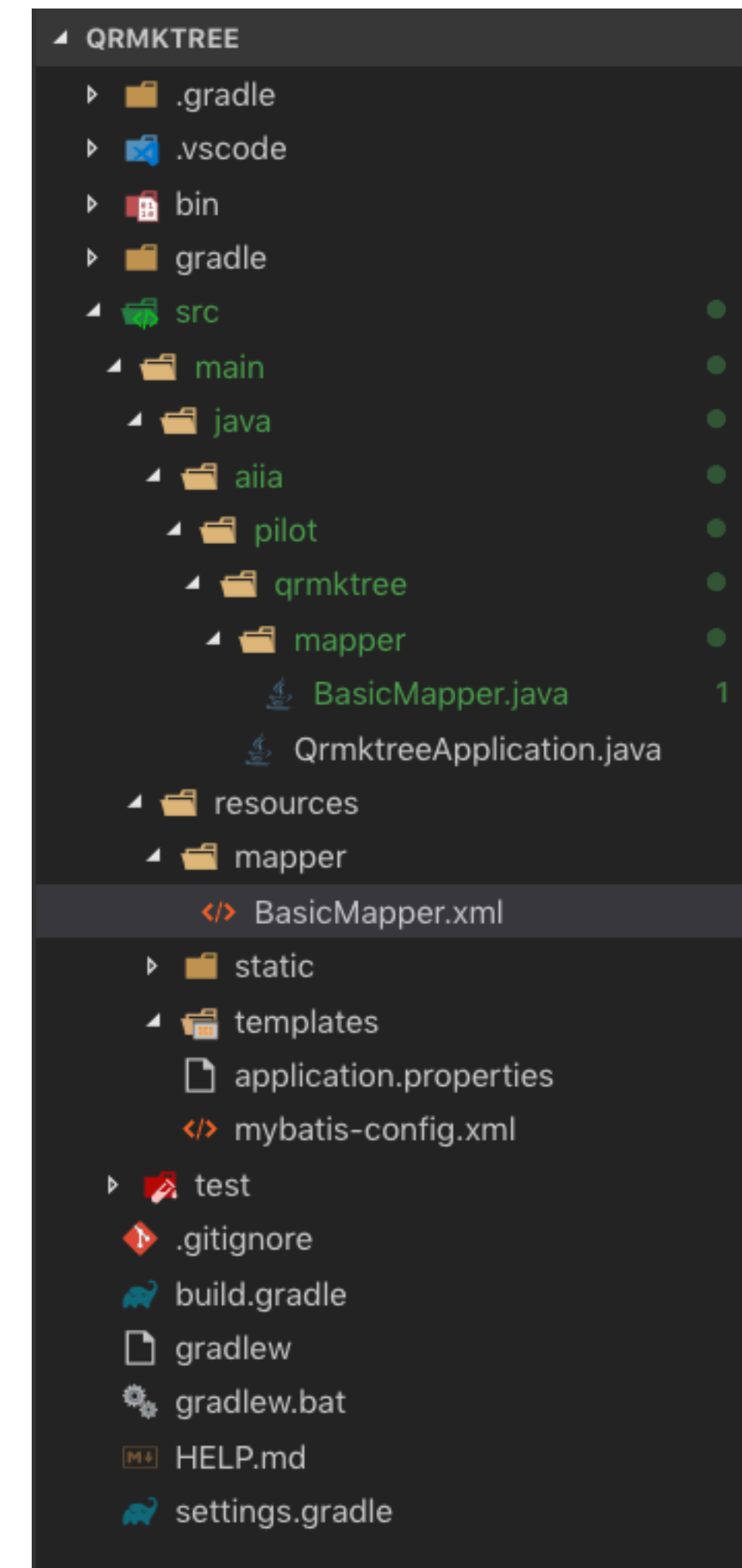
</configuration>
```

# 5. Mybatis configuration

오른쪽 구성과 같이 디렉토리와 파일을 생성한다.

1. qrmktree/mapper  
./BasicMapper.java

2.resource/mapper  
./BasicMapper.xml





# 5. Mybatis configuration

1. qrmktree/mapper  
./BasicMapper.java

```
package aiia.pilot.qrmktree.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Mapper;

@Mapper

public interface BasicMapper {

    List list();

}
```

# 5. Mybatis configuration

2.resource/mapper  
./BasicMapper.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="aiaa.pilot.qrmktree.mapper.BasicMapp
er">

    <select id="list" resultType="HashMap">

        SELECT * FROM member

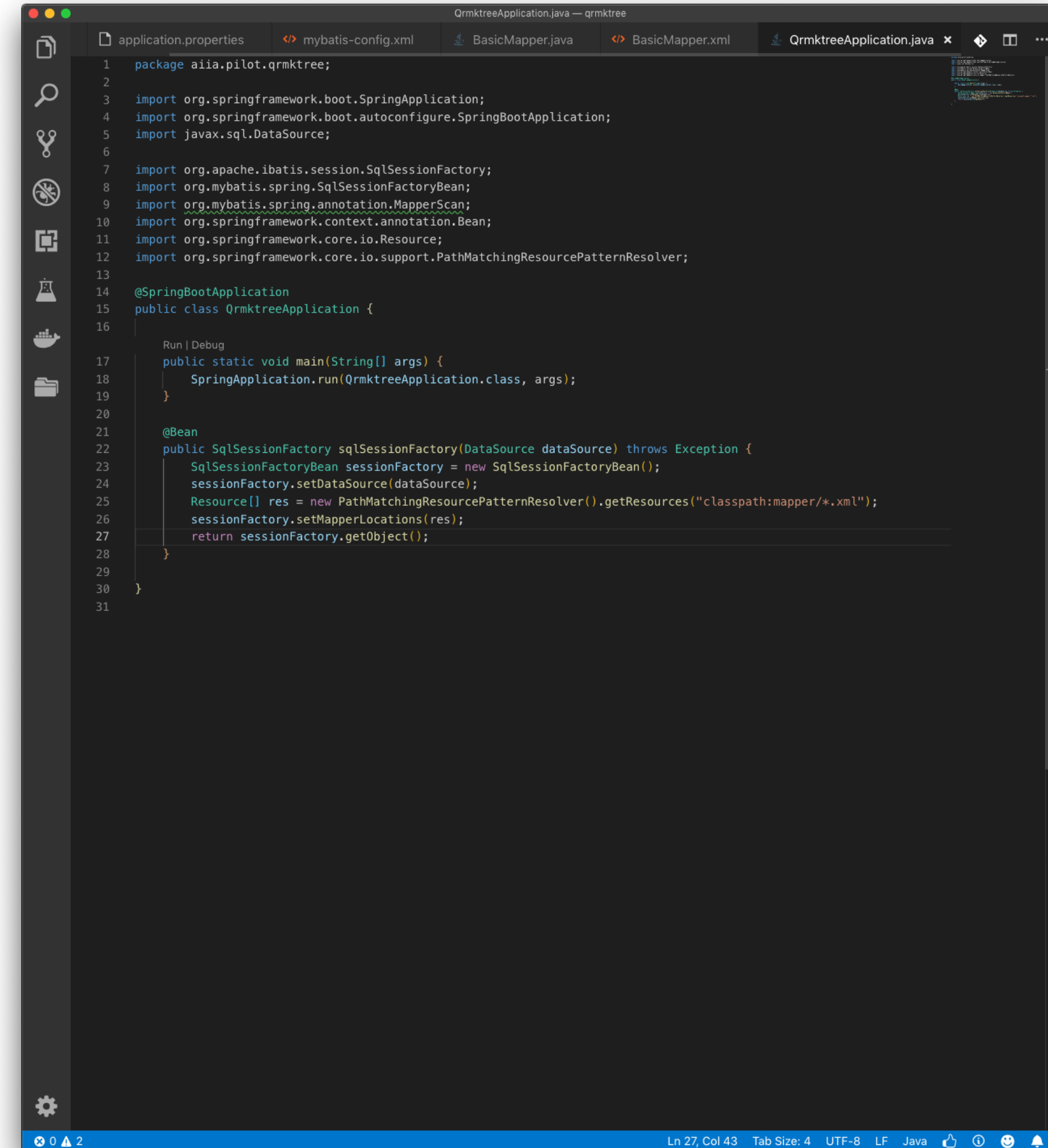
    </select>

</mapper>
```

# 5. Mybatis configuration

Mybatis는 sqlSession을 생성하기 위해 SqlSessionFactory를 사용한다. 따라서 이를 생성하고 유지하기 위해 아래 코드를 오른쪽의 위치에 삽입한다.

```
@Bean
public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws
Exception {
    SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean();
    sessionFactory.setDataSource(dataSource);
    Resource[] res = new
PathMatchingResourcePatternResolver().getResources("classpath:mapper/*.xml");
    sessionFactory.setMapperLocations(res);
    return sessionFactory.getObject();
}
```



```
1 package aia.pilot.qrmktree;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import javax.sql.DataSource;
6
7 import org.apache.ibatis.session.SqlSessionFactory;
8 import org.mybatis.spring.SqlSessionFactoryBean;
9 import org.mybatis.spring.annotation.MapperScan;
10 import org.springframework.context.annotation.Bean;
11 import org.springframework.core.io.Resource;
12 import org.springframework.core.io.support.PathMatchingResourcePatternResolver;
13
14 @SpringBootApplication
15 public class QrmktreeApplication {
16
17     Run | Debug
18     public static void main(String[] args) {
19         SpringApplication.run(QrmktreeApplication.class, args);
20     }
21
22     @Bean
23     public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {
24         SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean();
25         sessionFactory.setDataSource(dataSource);
26         Resource[] res = new PathMatchingResourcePatternResolver().getResources("classpath:mapper/*.xml");
27         sessionFactory.setMapperLocations(res);
28         return sessionFactory.getObject();
29     }
30 }
31
```

# 6. Controller 작성

.../qrmktree/controller 디렉토리를  
생성하고 HomeController.java를 만  
든 후 다음과 같은 코드를 삽입한다.

```
@RestController

public class HomeController {

    @Resource

    BasicMapper basicMapper;

    @GetMapping(value="/")

    public String home() {

        return "this is home";

    }

    @GetMapping(value="/api")

    public List getList() {

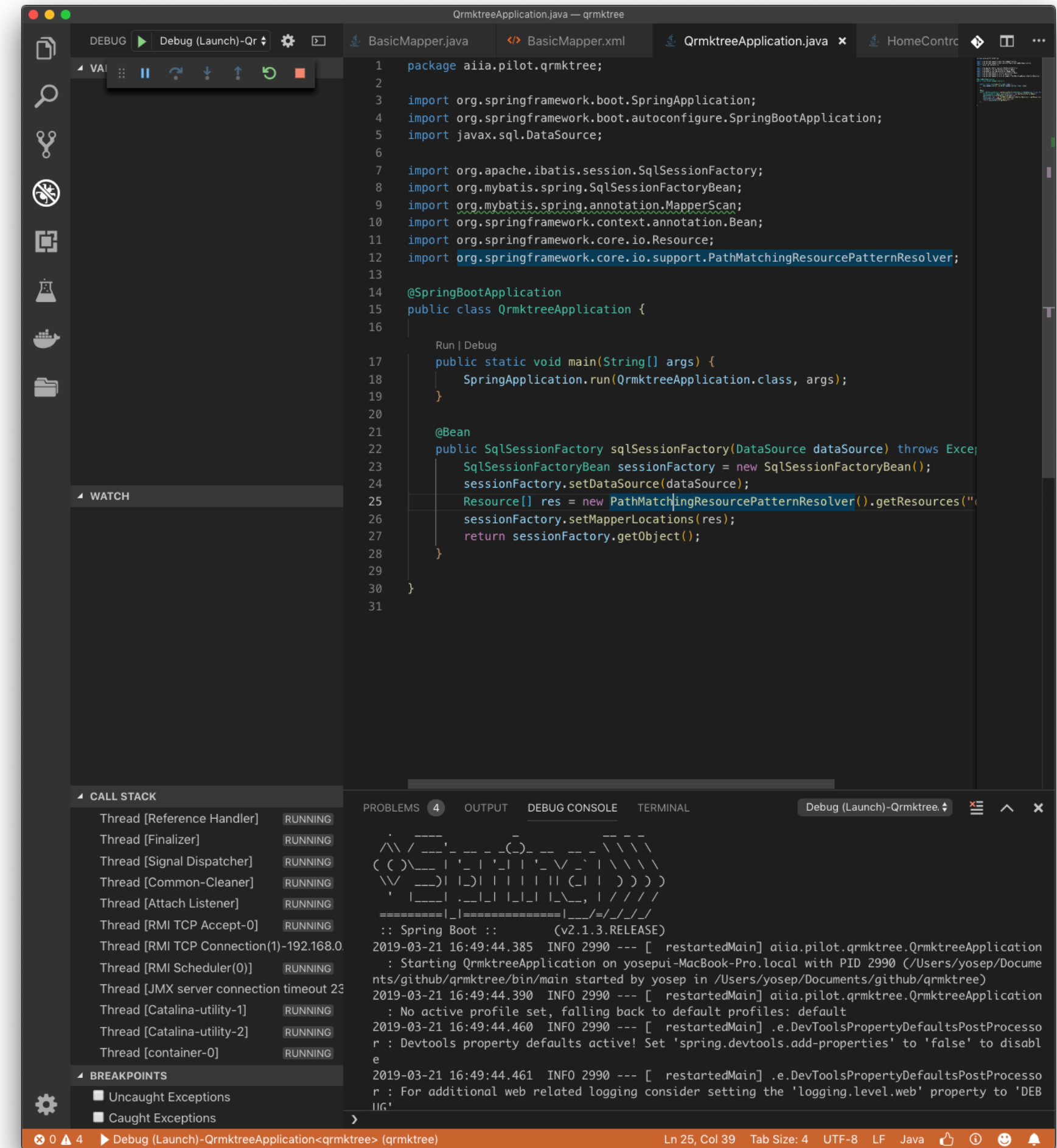
        return basicMapper.list();

    }

}
```

# 7. 실행

F5를 눌러 실행 Configuration을 확인  
하고(launch.json) 다시 한 번 F5를 눌  
러 Springboot에 내장되어있는  
Tomcat을 통해 서버 프로그램을 실행  
한다.



# 7. 실행

로컬 호스트에 접속하여 오른쪽과 같은  
실행 결과를 확인한다.

<https://localhost:8080/>

