3. Spring Boot中使用MyBatis

整合MyBatis之前，先搭建一个基本的Spring Boot项目开启Spring Boot。然后引入 `mybatis-spring-boot-starter` 和数据库连接驱动（这里使用关系型数据库Oracle 11g）。

mybatis-spring-boot-starter

在pom中引入：

XML

```xml
<dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
  <version>1.3.1</version>
</dependency>
```

不同版本的Spring Boot和MyBatis版本对应不一样，具体可查看官方文档：http://www.mybatis.org/spring-boot-starter/mybatis-spring-boot-autoconfigure/。

通过 `dependency:tree` 命令查看 `mybatis-spring-boot-starter` 都有哪些隐性依赖：

Bash

```
+- org.mybatis.spring.boot:mybatis-spring-boot-starter:jar:1.3.1:compile
|  +- org.springframework.boot:spring-boot-starter-jdbc:jar:1.5.9.RELEASE:compile
|  |  +- org.apache.tomcat:tomcat-jdbc:jar:8.5.23:compile
|  |  |  \- org.apache.tomcat:tomcat-juli:jar:8.5.23:compile
|  |  \- org.springframework:spring-jdbc:jar:4.3.13.RELEASE:compile
|  |     \- org.springframework:spring-tx:jar:4.3.13.RELEASE:compile
|  +- org.mybatis.spring.boot:mybatis-spring-boot-autoconfigure:jar:1.3.1:compile
|  +- org.mybatis:mybatis:jar:3.4.5:compile
|  \- org.mybatis:mybatis-spring:jar:1.3.1:compile
```

可发现其包含了 `spring-boot-starter-jdbc` 依赖，默认使用tomcat-jdbc数据源。

引入ojdbc6

由于版权的原因，我们需要将ojdbc6.jar依赖安装到本地的maven仓库，然后才可以在pom中进行配置。

下载ojdbc6.jar文件后，将其放到比较好找的目录下，比如D盘根目录。然后运行以下命令：

```Bash
C:\Users\Administrator>mvn install:install-file -Dfile=D:/ojdbc6.jar -DgroupId=com.oracle -DartifactId=ojdbc6 -Dversion=6.0 -Dpackaging=jar -DgeneratePom=true

...
[INFO] --- maven-install-plugin:2.4:install-file (default-cli) @ standalone-pom ---
[INFO] Installing D:\ojdbc6.jar to D:\m2\repository\com\oracle\ojdbc6\6.0\ojdbc6-6.0.jar
[INFO] Installing C:\Users\ADMINI~1\AppData\Local\Temp\mvninstall9103688544010617483.pom to D:\m2\repository\com\oracle\ojdbc6\6.0\ojdbc6-6.0.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.940 s
[INFO] Finished at: 2017-08-13T15:06:38+08:00
[INFO] Final Memory: 6M/145M
[INFO] ------------------------------------------------------------------------
```

接着在pom中引入：

```XML
<dependency>
  <groupId>com.oracle</groupId>
  <artifactId>ojdbc6</artifactId>
  <version>6.0</version>
</dependency>
```

这里的groupid就是你之前安装时指定的-Dgroupid的值，artifactid就是你安装时指定的-Dartifactid的值，version也一样。

引入mysql驱动

```xml
<!--使用mysql的数据驱动-->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.33</version>
</dependency>
```

修改一下durid当中的驱动以及数据库连接

```yaml
driver-class-name: com.mysql.cj.jdbc.Driver
# url: jdbc:oracle:thin:@localhost:1521:ORCL # 原作者的oracle的链接
url: jdbc:mysql://localhost:3306/test?useSSL=false&serverTimezone=UTC&characterEncoding=UTF-8
```

Druid数据源

Druid是一个关系型数据库连接池，是阿里巴巴的一个开源项目，地址：https://github.com/alibaba/druid。Druid不但提供连接池的功能，还提供监控功能，可以实时查看数据库连接池和SQL查询的工作情况。

配置Druid依赖

Druid为Spring Boot项目提供了对应的starter：

```xml
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>druid-spring-boot-starter</artifactId>
  <version>1.1.6</version>
</dependency>
```

Druid数据源配置

上面通过查看mybatis starter的隐性依赖发现，Spring Boot的数据源配置的默认类型是 `org.apache.tomcat.jdbc.pool.Datasource` ，为了使用Druid连接池，需要在application.yml下配置：

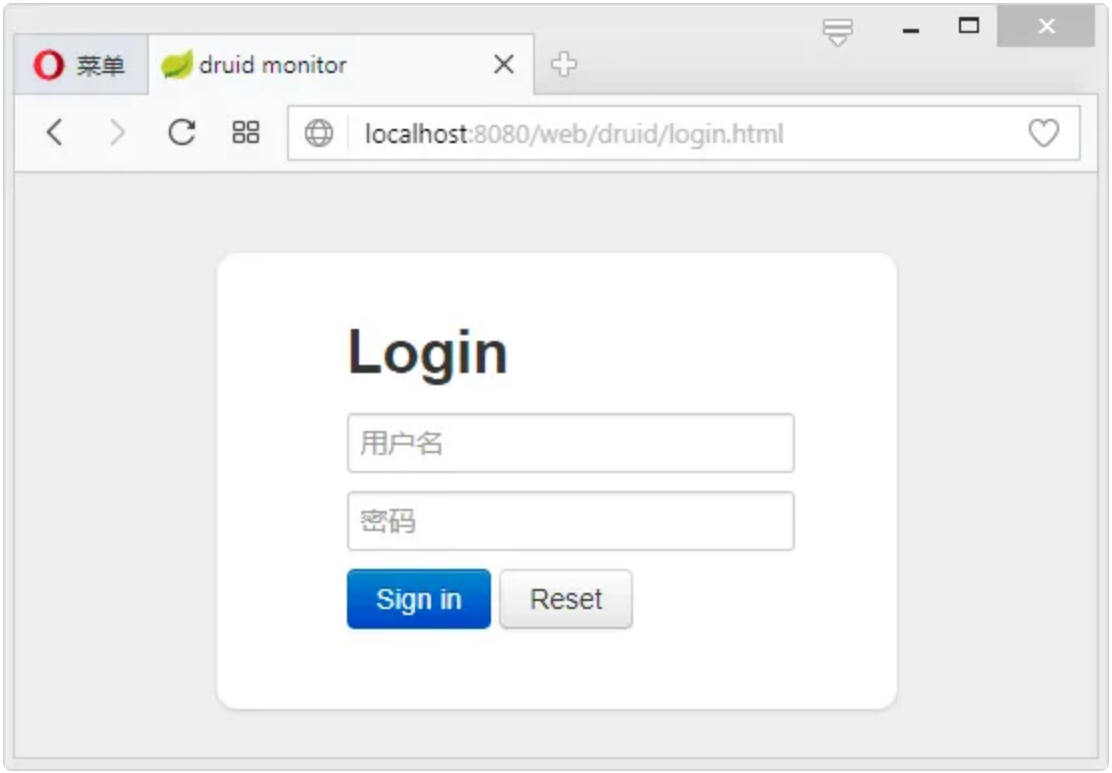
YAML

```yaml
server:
  context-path: /web

spring:
  datasource:
    druid:
      # 数据库访问配置, 使用druid数据源
      type: com.alibaba.druid.pool.DruidDataSource
      driver-class-name: oracle.jdbc.driver.OracleDriver
      url: jdbc:oracle:thin:@localhost:1521:ORCL
      username: scott
      password: 123456
      # 连接池配置
      initial-size: 5
      min-idle: 5
      max-active: 20
      # 连接等待超时时间
      max-wait: 30000
      # 配置检测可以关闭的空闲连接间隔时间
      time-between-eviction-runs-millis: 60000
      # 配置连接在池中的最小生存时间
      min-evictable-idle-time-millis: 300000
      validation-query: select '1' from dual
      test-while-idle: true
      test-on-borrow: false
      test-on-return: false
      # 打开PSCache，并且指定每个连接上PSCache的大小
      pool-prepared-statements: true
      max-open-prepared-statements: 20
      max-pool-prepared-statement-per-connection-size: 20
      # 配置监控统计拦截的filters, 去掉后监控界面sql无法统计, 'wall'用于防火墙
      filters: stat,wall
      # Spring监控AOP切入点，如x.y.z.service.*,配置多个英文逗号分隔
      aop-patterns: com.springboot.servie.*


      # WebStatFilter配置
      web-stat-filter:
        enabled: true
        # 添加过滤规则
        url-pattern: /*
        # 忽略过滤的格式
        exclusions: '*.js,*.gif,*.jpg,*.png,*.css,*.ico,/druid/*'

      # StatViewServlet配置
```

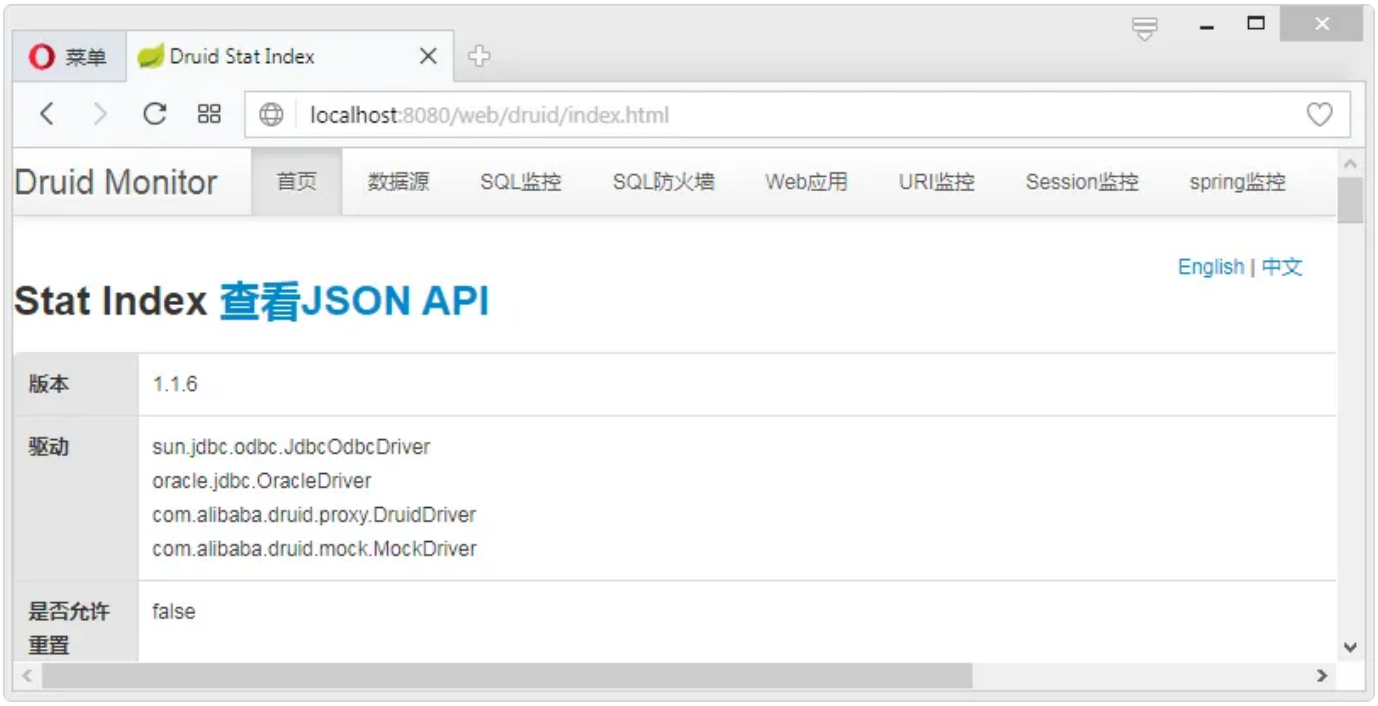
```yaml
      stat-view-servlet:
        enabled: true
        # 访问路径为/druid时，跳转到StatViewServlet
        url-pattern: /druid/*
        # 是否能够重置数据
        reset-enable: false
        # 需要账号密码才能访问控制台
        login-username: druid
        login-password: druid123
        # IP白名单
        # allow: 127.0.0.1
        #  IP黑名单（共同存在时，deny优先于allow）
        # deny: 192.168.1.218

      # 配置StatFilter
      filter:
        stat:
          log-slow-sql: true
```

上述配置不但配置了Druid作为连接池，而且还开启了Druid的监控功能。 其他配置可参考官方wiki——https://github.com/alibaba/druid/tree/master/druid-spring-boot-starter

此时，运行项目，访问http://localhost:8080/web/druid：



输入账号密码即可看到Druid监控后台：



关于Druid的更多说明，可查看官方wiki——

https://github.com/alibaba/druid/wiki/%E5%B8%B8%E8%A7%81%E9%97%AE%E9%A2%98

使用MyBatis

使用的库表：

PL/SQL

```
CREATE TABLE "SCOTT"."STUDENT" (
  "SNO" VARCHAR2(3 BYTE) NOT NULL ,
  "SNAME" VARCHAR2(9 BYTE) NOT NULL ,
  "SSEX" CHAR(2 BYTE) NOT NULL
);

INSERT INTO "SCOTT"."STUDENT" VALUES ('001', 'KangKang', 'M ');
INSERT INTO "SCOTT"."STUDENT" VALUES ('002', 'Mike', 'M ');
INSERT INTO "SCOTT"."STUDENT" VALUES ('003', 'Jane', 'F ');
```

创建对应实体：

```java
public class Student implements Serializable{
    private static final long serialVersionUID = -339516038496531943L;
    private String sno;
    private String name;
    private String sex;
    // get,set略
}
```

创建一个包含基本CRUD的StudentMapper：

```java
public interface StudentMapper {
    int add(Student student);
    int update(Student student);
    int deleteByIds(String sno);
    Student queryStudentById(Long id);
}
```

StudentMapper的实现可以基于xml也可以基于注解。

使用注解方式

继续编辑StudentMapper：

```java
@Component
@Mapper
public interface StudentMapper {
    @Insert("insert into student(sno,sname,ssex) values(#{sno},#{name},#{sex})")
    int add(Student student);

    @Update("update student set sname=#{name},ssex=#{sex} where sno=#{sno}")
    int update(Student student);

    @Delete("delete from student where sno=#{sno}")
    int deleteBysno(String sno);

    @Select("select * from student where sno=#{sno}")
    @Results(id = "student",value= {
        @Result(property = "sno", column = "sno", javaType = String.class),
        @Result(property = "name", column = "sname", javaType = String.class),
        @Result(property = "sex", column = "ssex", javaType = String.class)
    })
    Student queryStudentBySno(String sno);
}
```

简单的语句只需要使用@Insert、@Update、@Delete、@Select这4个注解即可，动态SQL语句需要使用@InsertProvider、@UpdateProvider、@DeleteProvider、@SelectProvider等注解。具体可参考MyBatis官方文档：http://www.mybatis.org/mybatis-3/zh/java-api.html。

使用xml方式

使用xml方式需要在application.yml中进行一些额外的配置：

```yaml
mybatis:
  # type-aliases扫描路径
  # type-aliases-package:
  # mapper xml实现扫描路径
  mapper-locations: classpath:mapper/*.xml
  property:
    order: BEFORE
```

测试

接下来编写Service：

```java
public interface StudentService {
    int add(Student student);
    int update(Student student);
    int deleteBysno(String sno);
    Student queryStudentBySno(String sno);
}
```

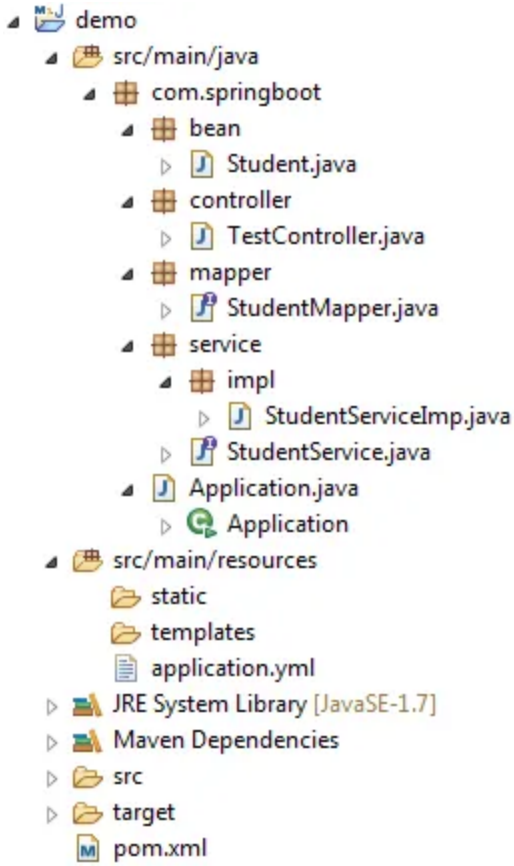
实现类：

```java
@Service("studentService")
public class StudentServiceImp implements StudentService{
    @Autowired
    private StudentMapper studentMapper;

    @Override
    public int add(Student student) {
        return this.studentMapper.add(student);
    }

    @Override
    public int update(Student student) {
        return this.studentMapper.update(student);
    }

    @Override
    public int deleteBysno(String sno) {
        return this.studentMapper.deleteBysno(sno);
    }

    @Override
    public Student queryStudentBySno(String sno) {
        return this.studentMapper.queryStudentBySno(sno);
    }
}
```

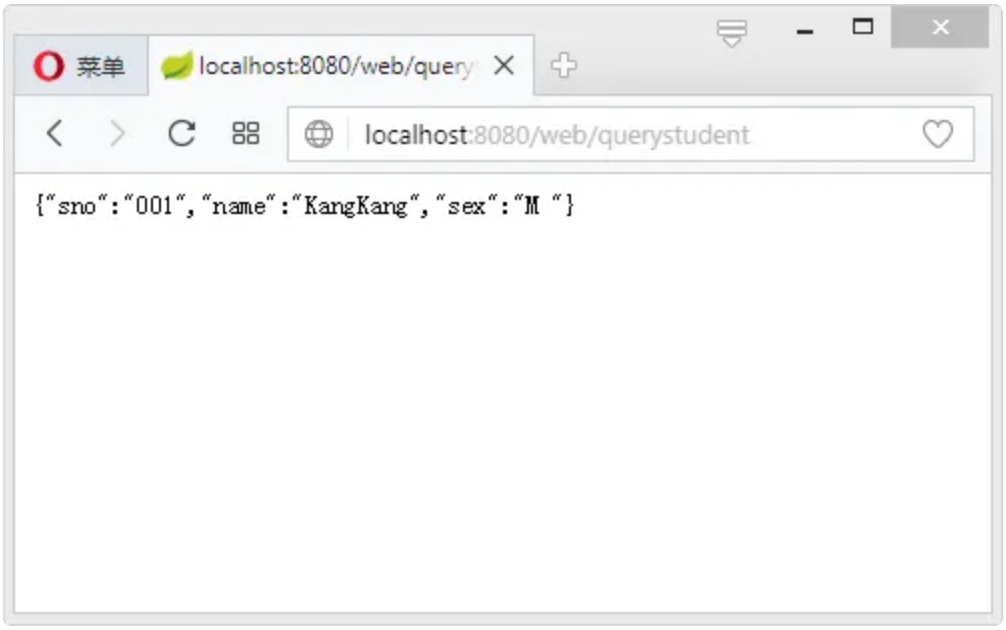
编写controller：

```java
@RestController
public class TestController {

    @Autowired
    private StudentService studentService;

    @RequestMapping( value = "/querystudent", method = RequestMethod.GET)
    public Student queryStudentBySno(String sno) {
        return this.studentService.queryStudentBySno(sno);
    }
}
```

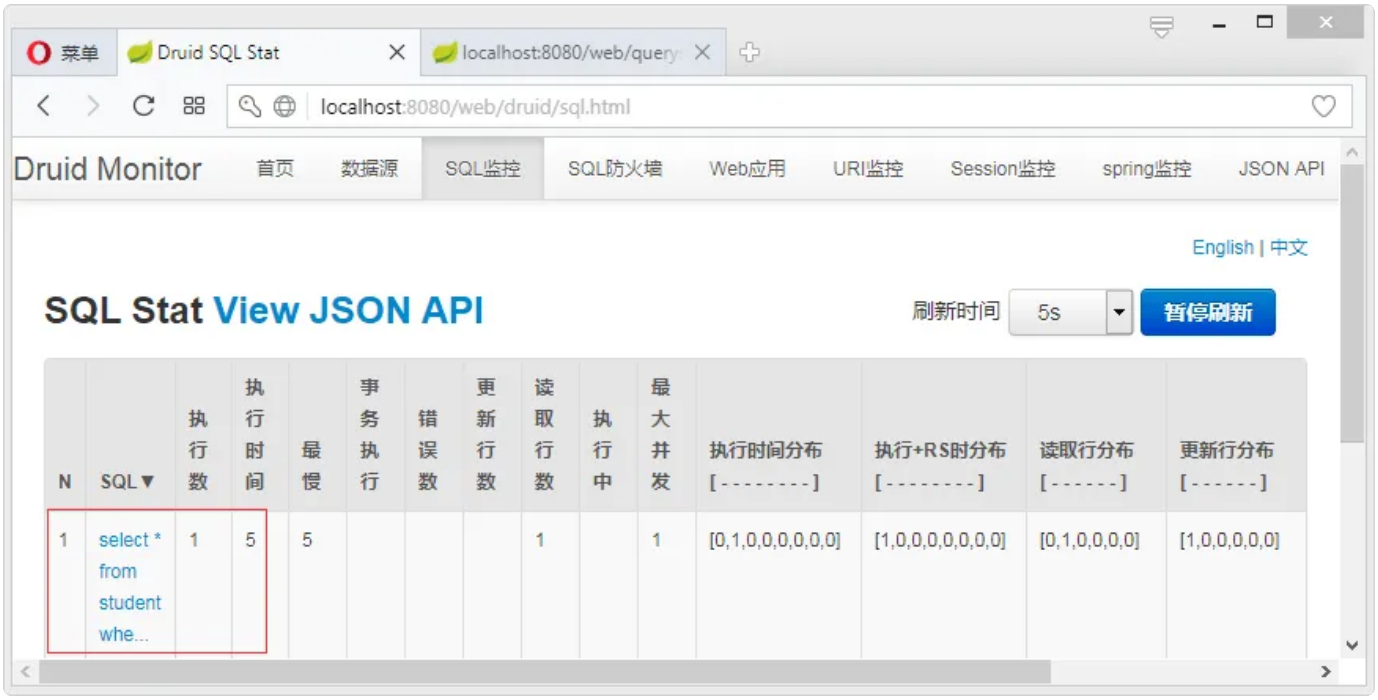
完整的项目目录如下图所示：



启动项目访问：http://localhost:8080/web/querystudent?sno=001：



查看SQL监控情况：



可看到其记录的就是刚刚访问/querystudent得到的SQL。

https://github.com/wuyouzhuguli/Spring-Boot-Demos/tree/master/03.Spring-Boot-MyBatis