链滴

# Spring Boot 快速入门之持久篇（三）

作者：yang17762622

原文链接：https://ld246.com/article/1568010662789

来源网站：链滴

# 一、前言

上一篇文章介绍了Spring Boot 的一些web相关的内容，咱们接下这边文章就来讲一下开发离不开的据。本篇以MySql+MyBatis为主。

# 二、整合mysql和mybatis

## 2.1 配置mysql和mybatis版本

```
<properties>
    <java.version>1.8</java.version>
    <mysql.version>5.1.44</mysql.version>
    <mybatis.version>3.4.5</mybatis.version>
    <mybatis.springboot.version>1.3.1</mybatis.springboot.version>
</properties>
```

## 2.2 添加依赖

```
<dependencies>
    <!-- mybatis -->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>${mybatis.version}</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-starter</artifactId>
        <version>${mybatis.springboot.version}</version>
```

```
    </dependency>
    <!-- mysql -->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>${mysql.version}</version>
    </dependency>
</dependencies>
```

## 2.3 配置数据连接

在 application-dev.properties 中添加以下配置，可以在不同配置文件配置对应的数据源，需要时切application.properties中的spring.profiles.active=dev即可。

```
#JDBC
spring.datasource.driverClassName = com.mysql.jdbc.Driver
spring.datasource.url = jdbc:mysql://127.0.0.1:3306/springboot?allowMultiQueries=true&use
nicode=true&characterEncoding=utf8&serverTimezone=Asia/Shanghai
spring.datasource.username = root
spring.datasource.password = Admin12345*
```

# 三、配置Generator

## 3.1 添加Generator插件依赖

在pom.xml中添加以下配置：

```
<plugin>
    <groupId>org.mybatis.generator</groupId>
    <artifactId>mybatis-generator-maven-plugin</artifactId>
    <version>1.3.3</version>
    <configuration>
      <verbose>true</verbose>
      <overwrite>true</overwrite>
    </configuration>
    <dependencies>
      <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java
      配置这个依赖主要是为了等下在配置MG的时候可以不用配置classPathEntry这样的一个属性，
免
        代码的耦合度太高 -->
      <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.38</version>
      </dependency>
    </dependencies>
</plugin>
```

## 3.2 新增generator配置文件

在resource文件夹下创建generatorConfig.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
       PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
       "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">

<generatorConfiguration>

    <context id="ssm" targetRuntime="MyBatis3">
        <plugin type="org.mybatis.generator.plugins.EqualsHashCodePlugin" />
        <plugin type="org.mybatis.generator.plugins.SerializablePlugin" />
        <plugin type="org.mybatis.generator.plugins.CaseInsensitiveLikePlugin" />
        <!-- <plugin type="org.mybatis.generator.plugins.ToStringPlugin"></plugin> -->
        <commentGenerator>
            <property name="suppressDate" value="true" />
            <property name="suppressAllComments" value="true" />
        </commentGenerator>

        <!-- 数据库连接 -->
        <jdbcConnection driverClass="com.mysql.jdbc.Driver"
                        connectionURL="jdbc:mysql://127.0.0.1:3306/springboot" userId="root" pass
ord="Admin12345*">
        </jdbcConnection>

        <javaTypeResolver>
            <property name="forceBigDecimals" value="false" />
        </javaTypeResolver>

        <!-- entity -->
        <javaModelGenerator targetPackage="com.y.springboot.model" targetProject="src/mai
/java">
            <property name="constructorBased" value="true" />
            <property name="enableSubPackages" value="true" />
            <property name="trimStrings" value="true" />
        </javaModelGenerator>

        <!--xml-->
        <sqlMapGenerator targetPackage="mybatis" targetProject="src/main/resources">
            <property name="enableSubPackages" value="true" />
        </sqlMapGenerator>

        <!--mapper.java-->
        <javaClientGenerator type="XMLMAPPER" targetPackage="com.y.springboot.mapper" t
rgetProject="src/main/java">
            <property name="enableSubPackages" value="true" />
        </javaClientGenerator>

        <!-- 需要生成的表  tableName(表名) domainObjectName(生成的java类名称)-->
        <table schema="mybatis" tableName="user" domainObjectName="User">
            <property name="constructorBased" value="true" />
            <property name="useActualColumnNames" value="false" />
            <property name="ignoreQualifiersAtRuntime" value="true" />
        </table>
    </context>
</generatorConfiguration>
```

# 四、测试

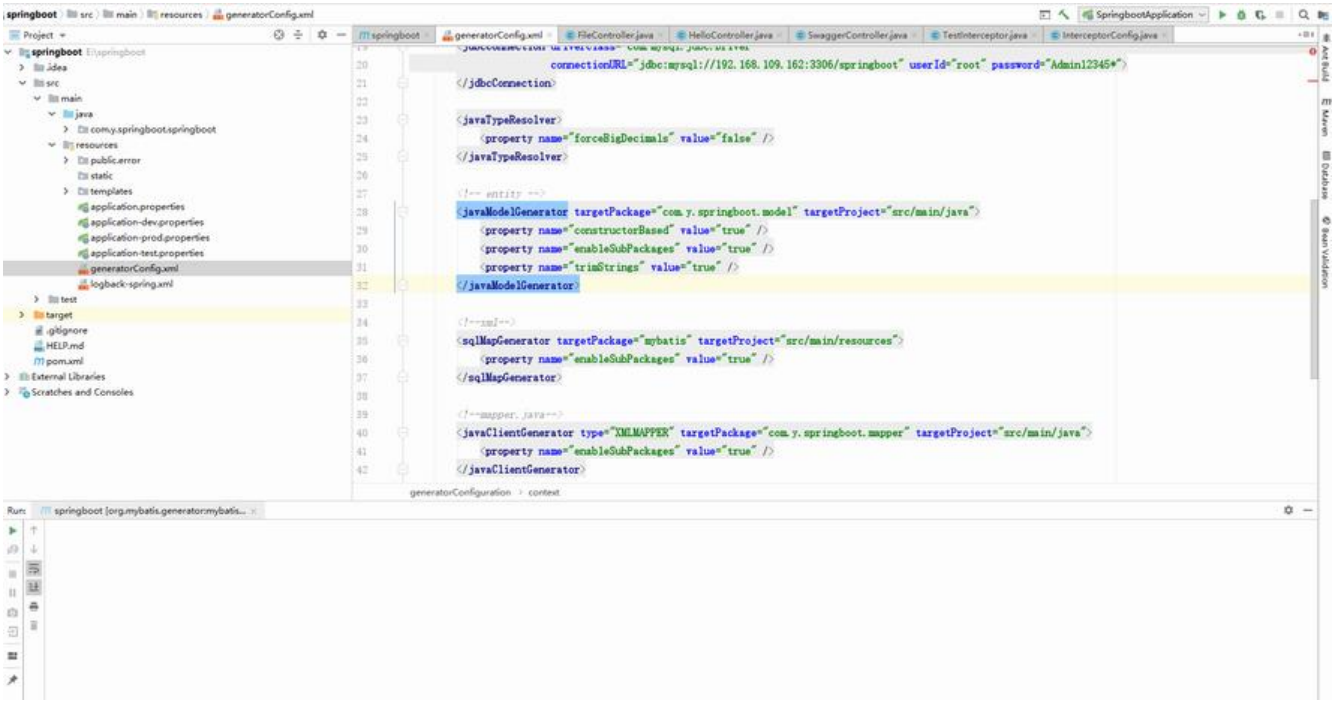
## 4.1 创建数据库表user

```
CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_name` varchar(255) DEFAULT NULL,
  `age` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

## 4.2 生成entity、mapper和xml

选择IDEA右侧Maven--> Plugins --> mybatis-generator -->mybatis-generator:generator

会将数据库表按照generatorConfig配置的路径名称生成对应的java实体类和xml方法



## 4.3 配置spring自动扫描和自动注入

在application.properties中添加：

```
#实体映射路径
#路径填写generatorConfig.xml中配置的路径
mybatis.mapper-locations = classpath:mybatis/*.xml
mybatis.type-aliases-package = com.tes.sys.com.healthsafe.model
```

修改SpringbootApplication

```
package com.y.springboot.springboot;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication(scanBasePackages = "com.y.springboot")
@MapperScan("com.y.springboot.mapper")
public class SpringbootApplication  {


    public static void main(String[] args) {
        SpringApplication.run(SpringbootApplication.class, args);
    }

}
```

## 4.4 测试类

TestService.java

```
package com.y.springboot.service;

import com.y.springboot.model.User;

public interface TestService {
    int addUser(User user);
}
```

TestServiceImpl.java

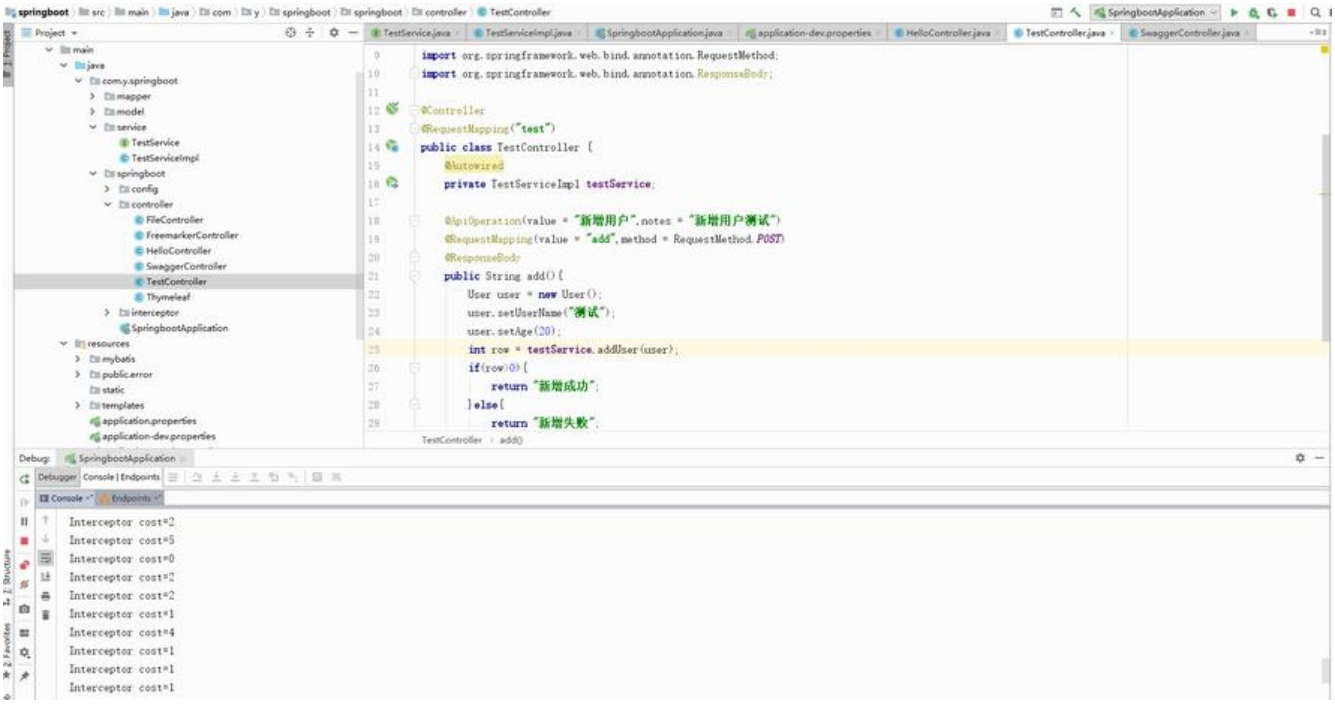
```
package com.y.springboot.service;

import com.y.springboot.mapper.UserMapper;
import com.y.springboot.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class TestServiceImpl implements TestService {
    @Autowired
    private UserMapper userMapper;
    @Override
    public int addUser(User user) {
        return userMapper.insertSelective(user);
    }
}
```

TestController.java

```
package com.y.springboot.springboot.controller;

import com.y.springboot.model.User;
import com.y.springboot.service.TestServiceImpl;
import io.swagger.annotations.ApiOperation;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping("test")
public class TestController {
    @Autowired
    private TestServiceImpl testService;

    @ApiOperation(value = "新增用户",notes = "新增用户测试")
    @RequestMapping(value = "add",method = RequestMethod.POST)
    @ResponseBody
    public String add(){
        User user = new User();
        user.setUserName("测试");
        user.setAge(20);
        int row = testService.addUser(user);
        if(row>0){
            return "新增成功";
        }else{
            return "新增失败";
        }
    }
}
```

测试效果图：




## 4.5 拓展Example的用法

本次使用generator生成的java实体会伴随一个对应的Example文件，可以用作去重/排序/分页/条件

询等操作

```
public void exampleTest(){
    UserExample userExample = new UserExample();
    //userName等于‘测试’且年龄等于20
    userExample.createCriteria().andUserNameEqualTo("测试").andAgeEqualTo(20);
    //按照年龄降序排序
    userExample.setOrderByClause("age desc");
    List<User> userList = userMapper.selectByExample(userExample);
}
```

更多详细Example操作请自行baidu~