链滴

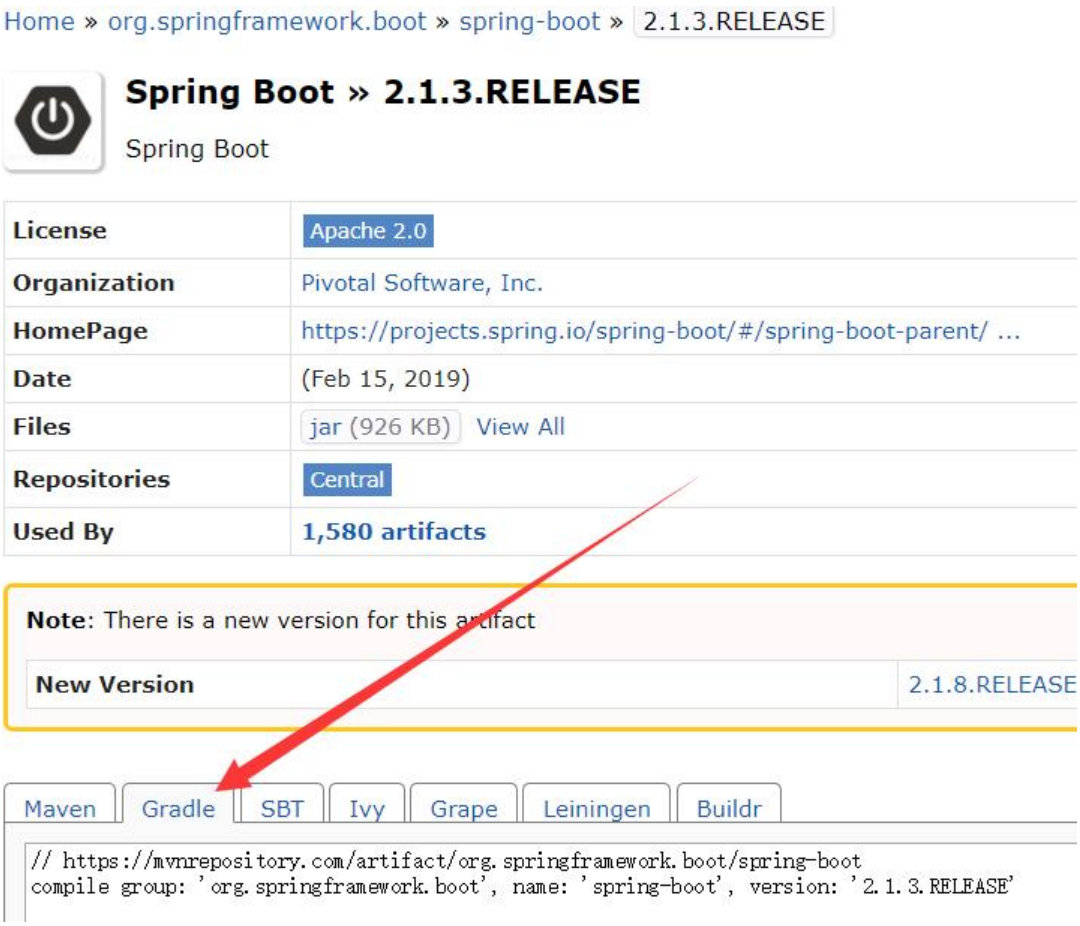
FirstGradle

作者：QForever

原文链接：https://ld246.com/article/1574148210153

来源网站：链滴

一、导入依赖



二、build.gradle

1. 整合SpringBoot

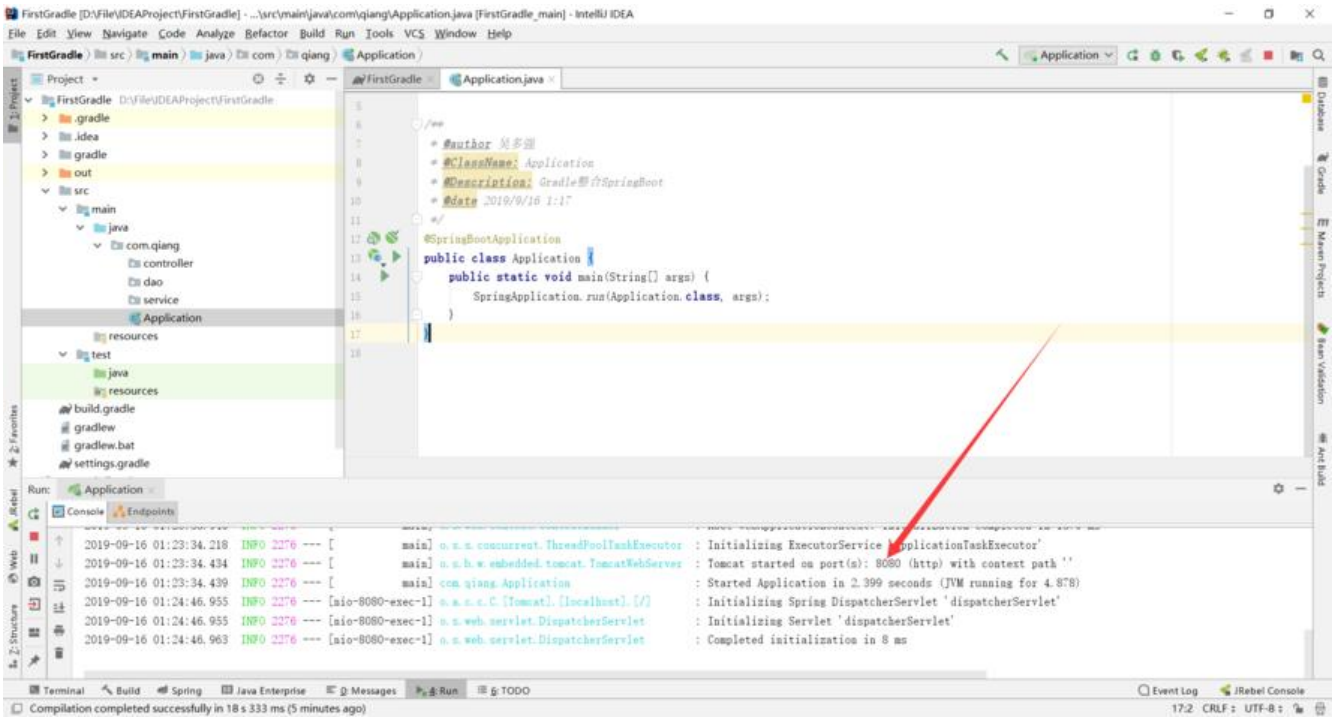
```
plugins {
    id 'java'
}

group 'com.qiang'
version '1.0.0-SNAPSHOT'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.12'
    compile group: 'org.springframework.boot', name: 'spring-boot-starter-web', version: '2.1.4
RELEASE'
}
```

原文链接：FirstGradle

三、Application

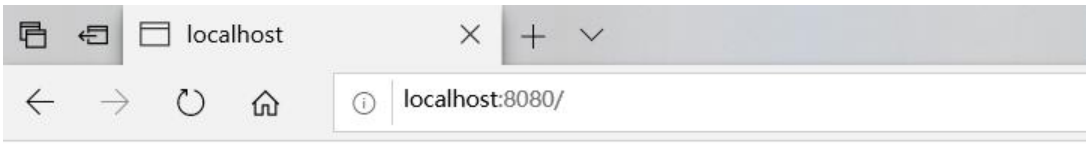
```
package com.qiang;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * @author 吴多强
 * @ClassName: Application
 * @Description: Gradle整合SpringBoot
 * @date 2019/9/16 1:17
 */
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

1. 启动项目



2. 启动成功

原文链接：FirstGradle



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Mon Sep 16 01:29:34 CST 2019
There was an unexpected error (type=Not Found, status=404).
No message available

四、整合Mabatis

1. 导入依赖

```
compile group: 'mysql', name: 'mysql-connector-java', version: '8.0.15'
compile group: 'com.alibaba', name: 'druid', version: '1.1.18'
compile group: 'org.mybatis.spring.boot', name: 'mybatis-spring-boot-starter', version: '2.0.1'
```

2. 设置资源文件夹

```
sourceSets {
    main {
        resources {
            srcDirs("src/main/java", "src/main/resources")
        }
    }
}
```

3. 在主函数上加上注解

```
@MapperScan({"com.qiang.mapper"})
```

五、application.yml

```
server:
  port: 8080
spring:
  application:
    name: FirstGradle
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=UTF-8&useSSL
false&serverTimezone=CTT
    username: root
    password: root
    type: com.alibaba.druid.pool.DruidDataSource
```

六、整合PageHelper

1. 导入依赖

```
compile group: 'com.github.pagehelper', name: 'pagehelper', version: '4.1.6'
```

2. 配置PageHelper

```
package com.qiang.config;

import com.github.pagehelper.PageHelper;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.Properties;

/**
 * @author 吴多强
 * @ClassName: MybatisConfig
 * @Description: PageHelper配置
 * @date 2019/9/16 16:37
 */
@Configuration
public class MybatisConfig {

    /**
     * 配置Mybatis的分页插件
     */
    @Bean
    public PageHelper pageHelper() {
        PageHelper pageHelper = new PageHelper();
        Properties properties = new Properties();
        properties.setProperty("offsetAsPageNum", "true");
        properties.setProperty("rowBoundsWithCount", "true");
        properties.setProperty("reasonable", "true");
        properties.setProperty("dialect", "mysql");
        pageHelper.setProperties(properties);
        return pageHelper;
    }
}
```

3. 使用

```
package com.qiang.controller;

import com.github.pagehelper.Page;
import com.github.pagehelper.PageHelper;
import com.qiang.service.UserService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.Resource;

/**
```

```
 * @author 吴多强
 * @ClassName: UserController
 * @Description: 用户控制器
 * @date 2019/9/16 1:51
 */
@RestController
public class UserController {
    private static final Logger logger = LoggerFactory.getLogger(UserController.class);
    @Resource
    private UserService userService;

    @RequestMapping("/user")
    public Object getAllUser() {
        Page<Object> page = PageHelper.startPage(0, 1);
        logger.info(page.toString());
        return userService.selectAll();
    }
}
```

七、整合Redis

1. 导入依赖

```
compile('org.springframework.boot:spring-boot-starter-data-redis:2.1.3.RELEASE') {
        exclude group: 'redis.clients', module: 'jedis'
        exclude group: 'io.lettuce', module: 'lettuce-core'
    }
    compile group: 'redis.clients', name: 'jedis', version: '2.9.0'
```

2. Redis配置

```
package com.qiang.config;

import com.fasterxml.jackson.annotation.JsonAutoDetect;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisStandaloneConfiguration;
import org.springframework.data.redis.connection.jedis.JedisClientConfiguration;
import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.Jackson2JsonRedisSerializer;
import org.springframework.data.redis.serializer.StringRedisSerializer;
import redis.clients.jedis.JedisPoolConfig;

/**
 * @author 吴多强
 * @ClassName: RedisConfig
 * @Description: Redis配置
 * @date 2019/9/16 16:49
```

```
 */
@Configuration
public class RedisConfig {

    /**
     * Redis服务器地址
     */
    private String host = "127.0.0.1";
    /**
     * Redis服务器连接端口
     */
    private int port = 6379;
    /**
     * Redis数据库索引（默认为0）
     */
    private int database = 15;
    /**
     * 连接池中的最大空闲连接
     */
    private int maxIdle = 8;
    /**
     * 连接池最大阻塞等待时间（使用负值表示没有限制）
     */
    private long maxWaitMillis = -1;
    /**
     * 连接池最大连接数（使用负值表示没有限制）
     */
    private int maxActive = 8;

    /**
     * 初始化
     */
    @Bean
    public JedisPoolConfig jedisPoolConfig() {
        JedisPoolConfig jedisPoolConfig = new JedisPoolConfig();
        jedisPoolConfig.setMaxTotal(maxActive);
        jedisPoolConfig.setMaxWaitMillis(maxWaitMillis);
        jedisPoolConfig.setMaxIdle(maxIdle);
        return jedisPoolConfig;
    }

    /**
     * 注入RedisConnectionFactory
     */
    @Bean
    public RedisConnectionFactory redisConnectionFactory(JedisPoolConfig jedisPoolConfig) {
        RedisStandaloneConfiguration redisStandaloneConfiguration = new RedisStandaloneCo
figuration();
        redisStandaloneConfiguration.setHostName(host);
        redisStandaloneConfiguration.setPort(port);
        redisStandaloneConfiguration.setDatabase(database);
        JedisClientConfiguration.JedisPoolingClientConfigurationBuilder jedisPoolConfigBuilder

                (JedisClientConfiguration.JedisPoolingClientConfigurationBuilder) JedisClientConfig
```

```
ration.builder();
        jedisPoolConfigBuilder.poolConfig(jedisPoolConfig);
        return new JedisConnectionFactory(redisStandaloneConfiguration);
    }

    @Bean
    public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory redisConnect
onFactory) {
        RedisTemplate redisTemplate = new RedisTemplate();
        redisTemplate.setConnectionFactory(redisConnectionFactory);
        Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson2JsonRedisSeriali
er(Object.class);
        ObjectMapper objectMapper = new ObjectMapper();
        // 设置任何字段可见
        objectMapper.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
        // 设置不是final的属性可以转换
        objectMapper.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
        jackson2JsonRedisSerializer.setObjectMapper(objectMapper);
        StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();
        // key采用String的序列化方式
        redisTemplate.setKeySerializer(stringRedisSerializer);
        // hash的key采用String的序列化方式
        redisTemplate.setHashKeySerializer(stringRedisSerializer);
        // value序列化方式采用jackson序列化方式
        redisTemplate.setValueSerializer(jackson2JsonRedisSerializer);
        // hash的value序列化方式采用jackson序列化方式
        redisTemplate.setHashValueSerializer(jackson2JsonRedisSerializer);
        redisTemplate.afterPropertiesSet();
        redisTemplate.setEnableTransactionSupport(true);
        return redisTemplate;
    }
}
```

3. RestTemplate配置

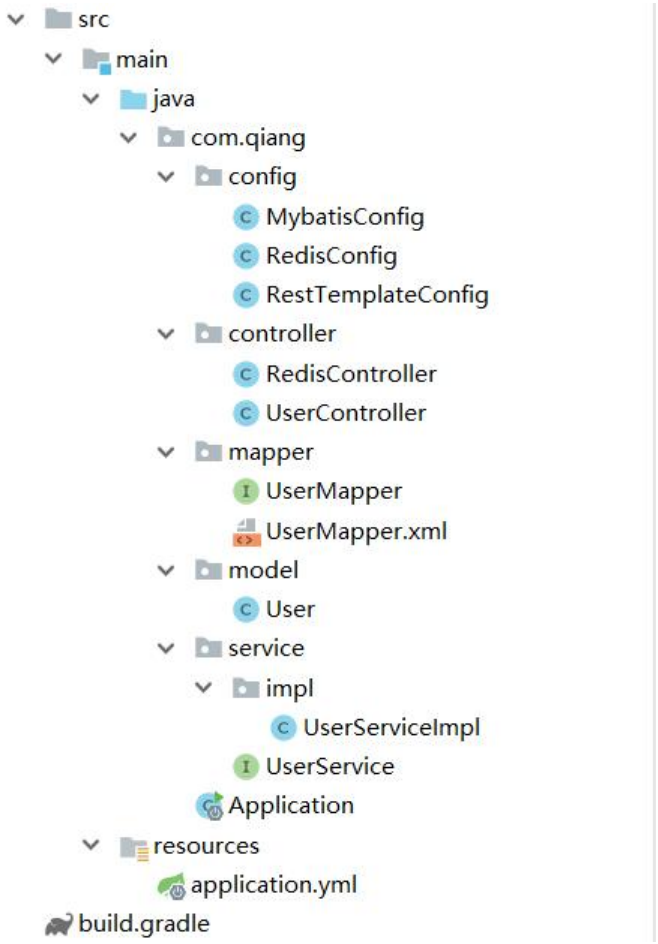
```
package com.qiang.config;

import org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.client.ClientHttpRequestFactory;
import org.springframework.http.client.SimpleClientHttpRequestFactory;
import org.springframework.web.client.RestTemplate;

/**
 * @author 吴多强
 * @ClassName: RestTemplateConfig
 * @Description: RestTemplate配置
 * @date 2019/9/16 17:25
 */
@Configuration
public class RestTemplateConfig {
    @Bean
```

```
    @ConditionalOnMissingBean({RestTemplate.class})
    public RestTemplate restTemplate(ClientHttpRequestFactory clientHttpRequestFactory) {
        return new RestTemplate(clientHttpRequestFactory);
    }

    @Bean
    @ConditionalOnMissingBean({ClientHttpRequestFactory.class})
    public ClientHttpRequestFactory clientHttpRequestFactory() {
        SimpleClientHttpRequestFactory factory = new SimpleClientHttpRequestFactory();
        factory.setReadTimeout(15000);
        factory.setConnectTimeout(15000);
        return factory;
    }
}
```

4. 使用

```
package com.qiang.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * @author 吴多强
 * @ClassName: RedisController
 * @Description:
 * @date 2019/9/16 18:01
 */
@RestController
public class RedisController {
    @Autowired
    private RedisTemplate redisTemplate;

    @RequestMapping("/redis")
    public Object getRedisValues(){
        redisTemplate.opsForValue().set("a","b");
        return redisTemplate.opsForValue().get("a");
    }
}
```

八、项目结构



九、源代码

FirstGradle.zip