1、pom.xml

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.2.5.RELEASE</version>

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.2.2</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.2.8</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat</groupId>

<artifactId>tomcat-jdbc</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.1.43</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-milestone</id>

<url>https://repo.spring.io/libs-release</url>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

<id>spring-milestone</id>

<url>https://repo.spring.io/libs-release</url>

</pluginRepository>

</pluginRepositories>

你实际在你本地去搭建这个工程的时候，你首先就会发现说，你一修改这个pom.xml，发现下载各种spring boot依赖包，下载巨慢巨慢

北京，宽带，100M，联通，还是下载的巨慢

手工下载依赖，并安装到本地maven仓库

（1）在maven中央仓库搜索jar包，如果没有找到，就得手动在百度里面找，下载jar下来

（2）根据jar对应的group id，artifact id，找到自己本地的maven仓库，对应的目录，将jar包拷贝到那个目录里面去

jmxtool，groupId=com.sun.jdmk，artifactId=jmxtools，version=1.2.1

com\sun\jdmk\jmxtools\1.2.1

（3）手工执行mvn install:install-file的命令，在本地仓库中安装这个依赖

mvn install:install-file -Dfile=E:\apache-maven-3.0.5\mvn\_repo\com\sun\jdmk\jmxtools\1.2.1\jmxtools-1.2.1.jar -DgroupId=com.sun.jdmk -DartifactId=jmxtools -Dversion=1.2.1 -Dpackaging=jar

（4）强制kill掉你的eclipse

（5）重新再进入eclips，这个时候肯定是会报很多的错误的，重新加载maven依赖

（6）反复循环，手工下载了，十几个到二十个依赖，然后最终所有的依赖全部成功下载到了本地，工程部报错

2、配置文件（src/main/resources）

Application.properties

server.port=8081

spring.datasource.url=jdbc:mysql://192.168.31.85:3306/eshop

spring.datasource.username=eshop

spring.datasource.password=eshop

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

说明：我已经在一个虚拟机中，安装好了一个mysql数据库，大家需要自己在自己本地安装一个mysql，配置好对应的url连接串，还有对应的用户名和密码就可以了

怎么安装mysql，大家自己网上查一下吧，java工程师，大学的学生，自己在本地安装一个mysql还是可以搞定的吧

mybatis/UserMappper.xml

templates/hello.html

3、Application

@EnableAutoConfiguration

@SpringBootApplication

@ComponentScan

@MapperScan("com.roncoo.eshop.cache.mapper")

public class Application {

@Bean

@ConfigurationProperties(prefix="spring.datasource")

public DataSource dataSource() {

return new org.apache.tomcat.jdbc.pool.DataSource();

}

@Bean

public SqlSessionFactory sqlSessionFactoryBean() throws Exception {

SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();

sqlSessionFactoryBean.setDataSource(dataSource());

PathMatchingResourcePatternResolver resolver = new PathMatchingResourcePatternResolver();

sqlSessionFactoryBean.setMapperLocations(resolver.getResources("classpath:/mybatis/\*.xml"));

return sqlSessionFactoryBean.getObject();

}

@Bean

public PlatformTransactionManager transactionManager() {

return new DataSourceTransactionManager(dataSource());

}

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

4、HelloController

5、完成两个服务的构建

《亿级流量电商详情页系统的大型高并发与高可用缓存架构实战》

龙果学院独家发布：<http://www.roncoo.com/>

课程学习地址：<http://www.roncoo.com/course/view/af7d501667fe4a19a60e9467e6d2b3d9>

1、亿级流量的电商网站的商品详情页系统架构

2、大型的企业级缓存架构，支撑高并发与高可用

3、几十万QPS的高并发+99.99%高可用+1T以上的海量数据+绝对数据安全的redis集群架构

4、高并发场景下的数据库+缓存双写一致性保障方案

5、大缓存的维度化拆分方案

6、基于双层nginx部署架构的缓存命中率提升方案

7、基于kafka+spring boot+ehcache+redis+nginx+lua的多级缓存架构

8、基于zookeeper的缓存并发更新安全保障方案

9、基于storm+zookeeper的大规模缓存预热解决方案

10、基于storm+zookeeper+nginx+lua的热点缓存自动降级与恢复解决方案

11、基于hystrix的高可用缓存服务架构

12、hystrix的进阶高可用架构方案、架构性能优化以及监控运维

13、基于hystrix的大规模缓存雪崩解决方案

14、高并发场景下的缓存穿透解决方案

15、高并发场景下的缓存失效解决方案