https://spring.io/guides/gs/async-method/

官方指南文档

**异步**

**如下模式，也可以在service内定义多个异步方法，将类的@Component换成@Service即可（其实就是将类加入到IOC容器内）方法加@Async注解**

import com.first.study.service.mybatis.dao.DepartHRMapper;  
import com.first.study.service.mybatis.dao.UserHRMapper;  
import com.first.study.service.mybatis.model.DepartHR;  
import com.first.study.service.mybatis.model.UserHR;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.**Autowired**;  
import org.springframework.scheduling.annotation.**Async**;  
import org.springframework.stereotype.**Component**;  
import org.springframework.stereotype.**Service**;  
import java.util.List;  
import static jodd.util.ThreadUtil.*sleep*;**@Component**public class TestThread {  
 Logger logger= LoggerFactory.*getLogger*(TestThread.class);  
 **@Autowired** DepartHRMapper departHRMapper;  
 **@Autowired** UserHRMapper userHRMapper;  
 **@Async**//注解，表示下面的方法将在单独的线程上运行  
 public void getUser(){  
 long start=System.*currentTimeMillis*();  
 List<UserHR> users=userHRMapper.getNS();  
 *sleep*(1000L);  
 long end=System.*currentTimeMillis*();  
 System.*err*.println(**"getUser"**+(end-start));  
 }  
 **@Async**//注解，表示下面的方法将在单独的线程上运行  
 public void getDepart(){  
 long start=System.*currentTimeMillis*();  
 List<DepartHR> departs=departHRMapper.getNS();  
 *sleep*(1000L);  
 long end=System.*currentTimeMillis*();  
 System.*err*.println(**"getDepart"**+(end-start));  
 }  
}

**启动类上加注解@EnableAsync**

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.**ServletComponentScan**;  
import org.springframework.cache.annotation.**EnableCaching**;  
import org.springframework.scheduling.annotation.**EnableAsync**;  
import org.springframework.scheduling.annotation.**EnableScheduling**;  
  
**@SpringBootApplication**(scanBasePackages={**"com.first.study.\*"**})  
**@MapperScan**(**"com.first.study.\*\*.dao"**)  
**@EnableAsync**public class Application {  
 //默认启动器  
 public static void main(String[] args) throws Exception {  
 SpringApplication.*run*(Application.class, args);  
 }  
}

启动加载类启动（类似controler调用）

import com.first.study.service.service.first.service.RedisService;  
import com.first.study.service.service.thread.TestThread;  
import org.springframework.beans.factory.annotation.**Autowired**;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.core.annotation.**Order**;  
import org.springframework.scheduling.annotation.**Async**;  
import org.springframework.stereotype.**Component**;**@Component**//添加进IOC容器  
**@Order**(value = 1)//设置启动优先级，越小优先级越高  
public class TestCommandLineRunner implements CommandLineRunner {  
 **@Autowired** RedisService redisService;  
 **@Autowired** TestThread testThread;  
 **@Override** public void run(String... args) throws Exception {  
 System.*err*.println(**"CommandLineRunner..."**);  
 //三条线程同步进行，run一条;getUser()一条;getDepart()一条  
 long start=System.*currentTimeMillis*();  
 testThread.getDepart();  
 testThread.getUser();  
 long end=System.*currentTimeMillis*();  
 System.*err*.println(**"test"**+(end-start));  
 }  
}

扩展：注意监控线程执行完成的情况，后续学习观察者模式怎么监控