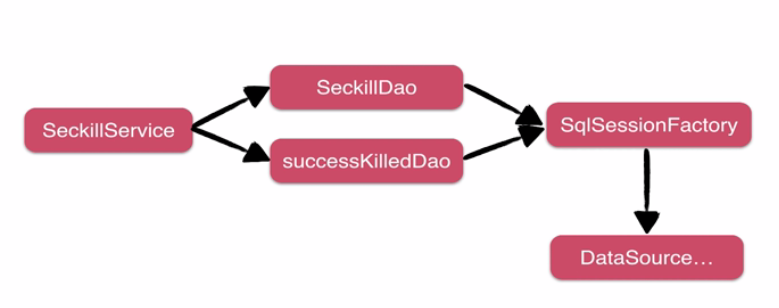
1. 业务对象依赖图(秒杀系统)



1. 为什么用IOC

对象创建统一托管

规范的生命周期管理

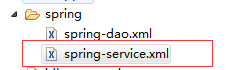
灵活的依赖注入

一致的获取对象：从容器中拿到任意已知对象的实例

1. Spring-IOC注入方式和场景



1. 代码实现
2. 新建一个spring-service.xml文件用来管理service依赖



1. 配置包扫描

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd"*>  <context:annotation-config />  <!-- 扫描service包下所有使用注解的类型 -->  <context:component-scan base-package=*"edu.hubu.seckill.service"*></context:component-scan>  </beans> |

1. 在service实现类声明bean,然后添加依赖注入(自动注入)

|  |
| --- |
| @Service  **public** **class** SeckillServiceImpl **implements** SeckillService {  **private** Logger logger = LoggerFactory.*getLogger*(**this**.getClass());    //注入service依赖  @Autowired  **private** SeckillDao seckillDao;    @Autowired  **private** SeckillSuccessDao seckillSuccessDao;    //MD5盐值字符串，用于混淆MD5  **private** **final** String slat = "sdasdsddsDERTTRTR#$%%DR";  **public** List<Seckill> getSeckillList() {  **return** seckillDao.queryAll(0, 4);    }  **public** Seckill getById(**long** seckillId) {  **return** seckillDao.queryById(seckillId);    }  **public** Exposer exportSeckillUrl(**long** seckillId) {  Seckill seckill = seckillDao.queryById(seckillId);  //如果秒杀不存在  **if**(seckill == **null**){  **return** **new** Exposer(**false**, seckillId);  }  Date start =seckill.getStartTime();  Date end = seckill.getEndTime();  Date now = **new** Date();  //秒杀开启或者已经结束  **if**(start.getTime() > now.getTime() || end.getTime() < now.getTime()){  **return** **new** Exposer(**false**, seckillId, now.getTime(),start.getTime(),end.getTime());  }  //转换特定字符串的过程，不可逆  String md5 = getMD5(seckillId); //**TODO**  **return** **new** Exposer(**true**, md5, seckillId);  }    //产生md5  **private** String getMD5(**long** seckillId){  String base = seckillId + "/" + slat;  String md5 = DigestUtils.*md5DigestAsHex*(base.getBytes());  **return** md5;  }  **public** SeckillExecution executeSeckill(**long** seckillId, **long** userPhone, String md5)  **throws** SeckillExceprion, SeckillCloseException, RepeatKillException {  **if**(md5 == **null** || !md5.equals(getMD5(seckillId))){  **throw** **new** SeckillExceprion("seckill data rewrite");  }  //执行秒杀逻辑:减库存，加记录秒杀行为  Date now = **new** Date();  //减库存  **try** {  **int** updateCount = seckillDao.reduceNumber(seckillId,now);  **if**(updateCount <= 0){  //没有更新记录，秒杀结束  **throw** **new** SeckillCloseException("seckill is closed");  }**else**{  //记录购买行为  **int** insertCount = seckillSuccessDao.insertSuccessKill(seckillId, userPhone);  **if**(insertCount <= 0){  **throw** **new** RepeatKillException("seckill repeated");  }**else**{  //秒杀成功  SeckillSuccess seckillSuccess = seckillSuccessDao.queryByIdWithSeckill(seckillId, userPhone);  **return** **new** SeckillExecution(seckillId, SeckillStatEnum.***SUCCESS***, seckillSuccess);  }  }  } **catch** (SeckillCloseException e1) {  **throw** e1;  }**catch** (RepeatKillException e2) {  **throw** e2;  }**catch** (Exception e) {  logger.error(e.getMessage(),e);  //所有编译期异常转换成运行期异常  **throw** **new** SeckillExceprion("seckill inner error:"+e.getMessage());  }  }  } |

在上面的代码中，说几点：

1. 写了一个关于获得MD5的方法，同时，定义MD5盐值字符串，用于混淆MD5
2. 写了一个枚举，专门存放秒杀状态，状态信息
3. 将所有的编译期的异常转换成了运行期异常