## Spring

1. 两种不同实现方式中，@Autowired的标注位置不同，它们都会在Spring在初始化userManagerImpl这个bean时，自动装配userDao这个属性，区别是：第一种实现中，Spring会直接将UserDao类型的唯一一个bean赋值给userDao这个成员变量；第二种实现中，Spring会调用 setUserDao方法来将UserDao类型的唯一一个bean装配到userDao这个属性。
2. @Resource（JSR-250标准注解，推荐使用它来代替Spring专有的@Autowired注解） 的作用相当于@Autowired，只不过@Autowired按byType自动注入，而@Resource默认按 byName自动注入罢了。@Resource有两个属性是比较重要的，分别是name和type，Spring将@Resource注解的name属性解析为bean的名字，而type属性则解析为bean的类型。所以如果使用name属性，则使用byName的自动注入策略，而使用type属性时则使用byType自动注入策略。如果既不指定name也不指定type属性，这时将通过反射机制使用byName自动注入策略。

@Resource装配顺序

如果同时指定了name和type，则从Spring上下文中找到唯一匹配的bean进行装配，找不到则抛出异常

如果指定了name，则从上下文中查找名称（id）匹配的bean进行装配，找不到则抛出异常

如果指定了type，则从上下文中找到类型匹配的唯一bean进行装配，找不到或者找到多个，都会抛出异常

如果既没有指定name，又没有指定type，则自动按照byName方式进行装配（见2）；如果没有匹配，则回退为一个原始类型（UserDao）进行匹配，如果匹配则自动装配 。

1. **Spring的事务传播特性：**

事务是会传播的”在Spring中有针对传播特性的多种配置我们大多数情况下只用其中的一种:PROPGATION\_REQUIRED：这个配置项的意思是说当我调用service层的方法的时候开启一个事务(具体调用那一层的方法开始创建事务，要看你的aop的配置),那么在调用这个service层里面的其他的方法的时候,如果当前方法产生了事务就用当前方法产生的事务，否则就创建一个新的事务。这个工作使由Spring来帮助我们完成的。

* 1. 只要定义为spring的bean就可以对里面的方法使用@Transactional注解。
  2. Spring的事务传播是Spring特有的。不是对底层jdbc的代理。
  3. 使用spring声明式事务，spring使用AOP来支持声明式事务，会根据事务属性，自动在[方法调用之前决定是否开启一个事务]，并在[方法执行之后]决定事务提交或回滚事务。
  4. Spring支持的PROPAGATION\_NESTED 与PROPAGATION\_REQUIRES\_NEW的区别:

PROPAGATION\_REQUIRES\_NEW：二个事务没有信赖关系，不会存在A事务的成功取决于B事务的情况。有可能存在A提交B失败。A失败（比如执行到doSomeThingB的时候抛出异常）B提交，AB都提交，AB都失败的可能。

PROPAGATION\_NESTED：与PROPAGATION\_REQUIRES\_NEW不同的是，内嵌事务B会信赖A。即存在A失败B失败。A成功，B失败。A成功，B成功。而不存在A失败，B成功。

* 1. 特别注意PROPAGATION\_NESTED的使用条件:使用JDBC 3.0驱动时,仅仅支持DataSourceTransactionManager作为事务管理器。需要JDBC 驱动的java.sql.Savepoint类。有一些JTA的事务管理器实现可能也提供了同样的功能。使用PROPAGATION\_NESTED，还需要把PlatformTransactionManager的nestedTransactionAllowed属性设为true;而 nestedTransactionAllowed属性值默认为false;
  2. 特别注意PROPAGATION\_REQUIRES\_NEW的使用条件：JtaTransactionManager作为事务管理器

1. AOP介绍：

Aspect(切面):指横切性关注点的抽象即为切面，它与类相似，只是两者的关注点不一样，类是对物体特征的抽象，而切面是横切性关注点的抽象。

Joinpoint(连接点):所谓连接点是指那些被拦截到的点。在spring中，这些点指的是方法，因为spring只支持方法类型的连接点，实际上Joinpoint还可以是field或类构造器)。

Pointcut(切入点):所谓切入点是指我们要对那些Joinpoint进行拦截的定义。

Advice(通知):所谓通知是指拦截到joinpoint之后所要做的事情就是通知.通知分为前置通知，后置通知，异常通知，最终通知，环绕通知。

Target(目标对象):代理的目标对象。

Weave(织入):指将aspects应用到target对象并导致proxy对象创建的过程称为织入。

Introduction(引入):在不修改类代码的前提下，Introduction可以在运行期为类动态地添加一些方法或Field。

1. GET请求和POST请求区别：根据HTTP规范，GET用于信息获取，而且应该是安全的和幂等的。所谓安全的意味着该操作用于获取信息而非修改信息。换句话说，GET 请求一般不应产生副作用。根据HTTP规范，POST表示可能修改变服务器上的资源的请求。GET请求的数据会附在URL之后（就是把数据放置在HTTP协议头中），以?分割URL和传输数据，参数之间以&相连。POST把提交的数据则放置在是HTTP包的包体中。GET方式提交的数据最多只能是1024字节，理论上POST没有限制，可传较大量的数据。
2. 缓存如何刷新： 1、定时刷新  2、主动刷新覆盖   ，每个缓存框架都有自带的刷新机制，或者说缓存失效机制，就拿Redis和 Ehcache举例， 他们都有自带的过期机制，另外主动刷新覆盖时，只需获取对应的key进行数据的覆盖即可。
3. Redis缓存如何保持同步：

* Slave端在配置文件中添加了slave of指令，于是Slave启动时读取配置文件，初始状态为REDIS\_REPL\_CONNECT。
* Slave端在定时任务serverCron(Redis内部的定时器触发事件)中连接Master，发送sync命令，然后阻塞等待master发送回其内存快照文件(最新版的Redis已经不需要让Slave阻塞)。
* Master端收到sync命令简单判断是否有正在进行的内存快照子进程，没有则立即开始内存快照，有则等待其结束，当快照完成后会将该文件发送给Slave端。
* Slave端接收Master发来的内存快照文件，保存到本地，待接收完成后，清空内存表，重新读取Master发来的内存快照文件，重建整个内存表数据结构，并最终状态置位为 REDIS\_REPL\_CONNECTED状态，Slave状态机流转完成。
* Master端在发送快照文件过程中，接收的任何会改变数据集的命令都会暂时先保存在Slave网络连接的发送缓存队列里（list数据结构），待快照完成后，依次发给Slave,之后收到的命令相同处理，并将状态置位为 REDIS\_REPL\_ONLINE。

1. SpringMVC接口解释

* DispatcherServlet：Spring提供的前端控制器，所有的请求都有经过它来统一分发。在DispatcherServlet将请求分发给Spring Controller之前，需要借助于Spring提供的HandlerMapping定位到具体的Controller。
* HandlerMapping：能够完成客户请求到Controller映射。
* Controller：需要为并发用户处理上述请求，因此实现Controller接口时，必须保证线程安全并且可重用。Controller将处理用户请求。一旦Controller处理完用户请求，则返回ModelAndView对象给DispatcherServlet前端控制器，ModelAndView中包含了模型（Model）和视图（View）。从宏观角度考虑，DispatcherServlet是整个Web应用的控制器；从微观考虑，Controller是单个Http请求处理过程中的控制器，而ModelAndView是Http请求过程中返回的模型（Model）和视图（View）。
* ViewResolver：Spring提供的视图解析器（ViewResolver）在Web应用中查找View对象，从而将相应结果渲染给客户。

1. SpringMVC运行原理：

* 客户端请求提交到DispatcherServlet
* 由DispatcherServlet控制器查询一个或多个HandlerMapping，找到处理请求的Controller
* DispatcherServlet将请求提交到Controller
* Controller调用业务逻辑处理后，返回ModelAndView
* DispatcherServlet查询一个或多个ViewResoler视图解析器，找到ModelAndView指定的视图
* 视图负责将结果显示到客户端

1. DispatcherServlet其主要工作有以下三项：

* 截获符合特定格式的URL请求。
* 初始化DispatcherServlet上下文对应的WebApplicationContext，并将其与业务层、持久化层的WebApplicationContext建立关联。
* 初始化Spring MVC的各个组成组件，并装配到DispatcherServlet中。