JAVA Spring

1. 框架特点：
   1. 轻量级：
      1. 兼容性
      2. 处理开销小
   2. 控制反转IOC：
      1. 促进了低耦合
      2. 用BEAN来创建一个个对象，到需要这些对象依赖时自动注入

3.面向切面（AOP）：一般的OOP时纵向的元素，而AOP关注的是横向的内容，横切关注点，横切关注面

4.容器：SPRING也类似于一个容器，可以配置BEAN的创建，数据库的交互，前后端的交互什么的。

5.Spring作为框架，可以把应用逻辑留给开发者。

1. Spring核心组件：
   1. DATA ACCESS
   2. WEB
   3. AOP
   4. Core Container
   5. Test
2. Spring常用模块：
   1. 核心容器：BeanFactory，用工厂模式实现了控制反转，将应用程序的配置和依赖性规范与实际的应用代码分开
   2. Spring上下文：是一个配置文件，为spring框架提供了上下文信息。
   3. Spring Aop：将面向切面的变成功能集成到了spring框架里，提高了复用性和管理的便捷性
   4. Spring Dao：为管理异常和处理错误信息做了处理，减少了各种错误处理。
   5. Spring Orm：提供了ORM的对象关系工具，其中包括了JDO，Hibernate和iBatis SQL Map，遵循了Spring通用事务和DAO异常层次结构
   6. Spring web 模块：简化了处理请求和绑定参数到域对象的工作。
   7. Spring Mvc框架：是构建web应用程序的MVC实现。
3. Spring主要包：
4. Spring常用注解：
   1. 类上面的注解：
      1. Controller：这个注解标记了他所标记的类是不是SpringMvc Controller对象，分发处理器会检测方法是否使用了RequestMappering注解
      2. Service：标注业务层主件
      3. RestController：实现了controller和responsebody的组合效果
      4. Component：组件不好归类时，可以用各种注解批注
      5. Repository：标记Dao层
5. controller中的注解：
   * 1. Responsebody：这样的话返回的不是html页面，而是其他格式的数据（json，xml）
     2. requestMappering：用来标注对应的路径
     3. Autowired：通过这个来消除set，get方法
     4. PathVariable：映射url中的变量
     5. Requestparam：获取参数
6. 其他：
   * 1. ModelAttribute
     2. SessionAttributes
     3. Vaild
     4. CookieValue
7. Spring 第三方结合：
   1. 权限：shiro
   2. 缓存：Ehcache：纯java进程内的缓存框架

Redis：一个开源的使用c语言编写的，基于内存可持久化的日志型，缓存，Key-Value数据库

1. 持久层框架：
   * 1. Hibernate：ORM，关系映射框架，为JDBC做了轻量级的对象封装，将POJO和数据库表建立映射关系，是一个全自动的orm框架
     2. Mybatis：是支持SQL查询，存储，映射的持久层框架，半ORM
2. 定时任务：
3. 校验框架：
   * 1. Hibernate vaildator：
        1. 验证bean字段，基于注解，快速高效。

Spring IOC原理（记得去看源码）：

1. 概念上spring ioc是一种控制反转的死项，意味着把设计好的对象交给容器控制，而不是由具体的类来创建，从而达到再使用时再注入的效果。
2. 实现原理上：java反射机制+xml
   * 1. 他步骤上是先初始化容器，再解析包下面的class的注解信息，再封装类反射后实例化对象，再用hashmap的形式保存类对象，到需要使用的时候再实例化对应的对象
     2. 源码：

<https://blog.csdn.net/nuomizhende45/article/details/81158383>

* + 1. 注入方法：
    2. 自动装配：

Spring AOP 原理：

1. 横切的技术，是把影响了多个类的公共行为封装到一个可重用模块，将他命名为切面，就是把和业务无关，却被业务共同调用的逻辑，责任封装，减少可操作性，可维护性。
2. 主要的应用场景：
   1. 权限
   2. 缓存
   3. 内容传递
   4. 错误处理
   5. 懒加载
   6. 调试
   7. 记录跟踪，优化，校准
   8. 性能优化
   9. 持久化
   10. 资源池
   11. 同步
   12. 事务
3. 核心概念：
   1. 切面是对横切关注点的抽象
   2. 横切关注点：对哪些方法进行拦截，拦截后怎么处理
   3. 织入：将切面应用到目标对象并导致代理对象创建的过程
   4. 引入：不改代码的前提下，再运行期为类动态地添加一些方法和字段
4. AOP代理的方式：
   * 1. JDKProxy代理
     2. Cglib代理

SpringMvc原理：

1. 围绕着DispatcherServlet来设计的：
   * 1. 接收浏览器的http请求
     2. 寻找处理器
     3. 调用处理器（controller）
     4. 具体业务
     5. ModelAndView，得到处理结果
     6. 处理视图映射
     7. 将model模型数据传给View展示
     8. 把View的jsp响应回浏览器

SpringBoot原理：

1. 创建独立spring应用程序
2. 嵌入式的Tomcat，无需部署WAR文件
3. 简化Maven配置
4. 自动配置Spring
5. 没有代码生成
6. 提供生产就绪功能

JPA原理：

1. 事务：保证了用户操作的ACID：原子性，一致性，隔离性，持久性
2. 事务分类：
   * 1. 本地事务：
        1. 获得数据库连接
        2. 执行事务命令
        3. 提交事务
        4. 失败报错会回滚
3. 分布式事务，采用userTx来实现对分布式事务管理的统一接口
4. 两阶段提交：
   * 1. 准备阶段
     2. 提交阶段

尽可能晚地提交事务，让事务在提交前尽可能地完成所有能完成的操作

Mybatis缓存：

1. 一级缓存是SqlSession级别的缓存，就是在同一个SqlSession中进行相同的SQL语句查询时，二级缓存是跨SqlSession的缓存，是Mapper级别的。
2. 一级缓存原理：
   * 1. 第一次发出查询的SQL时，查询结果写入SqlSession的一级缓存中，缓存使用的数据结构是一个Map
     2. 如果中间有commit操作，就清空
3. 二级缓存原理：
   * 1. 类似cache，需要自己配置，mapper

Tomcat结构：