Spring

1.框架优点

- 轻量级（不需要太多依赖）：Spring在大小和透明性方面绝对属于轻量级的，基础版本的Spring框架大约只有2MB。
- 控制反转(IOC)：Spring使用控制反转技术实现了松耦合。依赖被注入到对象，而不是创建或寻找依赖对象。
- 面向切面编程(AOP)： Spring支持面向切面编程，同时把应用的业务逻辑与系统的服务分离开来。
- 容器：Spring包含并管理应用程序对象的配置及生命周期。
- MVC框架：Spring的web框架是一个设计优良的web MVC框架，很好的取代了一些web框架。
- 事务管理：Spring对下至本地业务上至全局业务(JAT)提供了统一的事务管理接口。
- 异常处理：Spring提供一个方便的API将特定技术的异常(由JDBC, Hibernate, 或JDO抛出)转化为一致的、Unchecked异常。

2.Spring是一个分层的一站式框架

spring针对javaee的三层结构，每一层都提供了解决技术

web层：SpringMVC

业务层（Service层）：Bean管理（IOC），Spring声明式事务

持久层（Dao层）：Spring的JDBC模板、ORM模板用于整合其他的持久层框架

3.Spring最核心的两部分

- Aop：面向切面编程，扩展功能不是修改源代码实现
- Ioc: 控制反转，比如有有一个类，想要调用类里面的非静态方法，把对象的创建不是通过new方式实现，而是交给spring配置创建类对象

4.Spring整合web项目

为了防止每次访问时，都加载Spring配置文件

解决方案：把加载配置文件和创建对象过程在服务器启动时完成

使用ServletContext对象和监听器来实现
具体使用：

- 在服务器启动时，为每个项目创建一个ServletContext对象
- 在ServletContext对象创建时候，可以使用监听器监听到
- 当监听到的时候，加载Spring配置文件，应用配置文件创建对象
- 把创建好的对象放到ServletContext域对象里面（setAttribute方法）
- 获取对象时候，到ServletContext域得到 （getAttribute方法）

5.SpringMVC运行机制

首先有DispatcherServlet进行分发，按照HandlerMapping进行分发到具体的Controller上，然后调用Service服务层获取model，最后将model返回给前端渲染。

- 向服务器发送Http request请求，请求被**前端控制器（DispatcherServlet）**捕获。
- 前端控制器根据xml文件中的配置（或者注解）对请求的URL进行解析，得到请求资源标识符（URI）。然后根据该URI，调用**处理器映射器（HandlerMapping）**获得处理该请求的Handler以及Handler对应的拦截器，最后以 HandlerExecutionChain 对象的形式返回。
- 前端控制器根据获得的Handler，选择一个合适的**处理器适配器（HandlerAdapter）**去执行该Handler。
- 处理器适配器提取request中的模型数据，填充Handler入参，执行**处理器（Handler）**（也称之为Controller）.
- Handler(Controller)执行完成后，向处理器适配器返回一个**ModelAndView**对象，处理器适配器再向前端控制器返回该ModelAndView对象（ModelAndView只是一个逻辑视图）。
- 根据返回的ModelAndView，前端控制器请求一个适合的**视图解析器（ViewResolver）**（必须是已经注册到Spring容器中的ViewResolver）去进行视图解析，然后视图解析器向前端控制器返回一个真正的视图View（jsp）。

- 前端控制器通过Model解析出ModelAndView中的参数进行解析，最终展现出完整的View并通过Http response返回给客户端。

6.Spring注解

1）@Component：注解可以放在类的头上，将这个类交给Spring管理。

@Component不推荐使用。Comeponent的衍生注解：

@Service：对应的是业务层Bean

@controller:web层:由DispatcherServlet分发的请求，它把用户请求的数据经过业务处理层处理之后封装成一个Model，然后再把该Model返回给对应的View进行展示

@Resposity:Dao层

2）属性注入的注解：

@Autowired：对象类型的注入，按照类型进行注入

@Qualifier 和 @Autowired：一起用按照属性进行注入

@Value：设置普通属性的值

@Resource：和@Autowired注解都是用来实现依赖注入的

只是@Autowired按byType自动注入，而@Resource默认按 byName自动注入。

3）生命周期的注解：

@PostConstruct：初始化方法

@PreDestroy：销毁方法

4）作用域注解：

@scope

5）Controller相关的注解

Controller中有@RestController、@RequestMapping的类注解，然后成员上有@Autowired注解，然后方法上还有@GetMapping或者

@RequestMapping注解。

@RequestMapping：可以声明到类或方法上。用来处理请求地址映射的注解

@RequestParam：完成参数绑定的一个注解

@SessionAttributes：作用在方法上或者方法的参数上，将被注解的方法的返回值或者是被注解的参数作为Model的属性加入到Model中，然后Spring框架

自会将这个Model传递给ViewResolver。
@ModelAttribute：绑定请求参数到指定对象
@Required：用来检查bean在初始化时其声明的set方法是否被执行

7.Spring中用到的设计模式

- 代理模式—在AOP和remoting中被用的比较多。
- 单例模式—在spring配置文件中定义的bean默认为单例模式。
- 模板模式—用来解决代码重复的问题. 比如. RestTemplate, JmsTemplate, JpaTemplate。
- 工厂模式—BeanFactory用来创建对象的实例。
- Builder模式 - 自定义配置文件的解析bean是时采用builder模式，一步一步地构建一个beanDefinition
- 策略模式： Spring 中策略模式使用有多个地方，如 Bean 定义对象的创建以及代理对象的创建等。这里主要看一下代理对象创建的策略模式的实现。前面已经了解 Spring 的代理方式有两个 **Jdk 动态代理**和**CGLIB 代理**。这两个代理方式的使用正是使用了策略模式。

8.拦截器的使用

实现了接口HandlerInterceptor，在springmvc配置之中去注册拦截器。并设置哪些页面会被拦截。

- preHandle方法：controller调用之前用的。登录拦截、权限认证等等
- postHandle方法：controller执行之后，在返回model之前。：设置或者清理页面共用参数等等
- afterCompletion方法：controller执行之后。处理异常、记录日志