Spring学习

1. StopWatch类：

方便地提供计算任务耗时的功能

stopWatch.start ---> stopWatch.stop ---> stopwatch.prettyPrint

01428 091% method1

00138 009% method2

2. InitializingBean接口：

InitializingBean接口为bean提供了初始化方法的方式，它只包括afterPropertiesSet方法，凡是继承该接口的类，在初始化bean的时候会执行该方法。也可以在xml中写”init-method”指定初始化函数。

AfterPropertiesSet是在源码中直接调用的，init-method则是要通过反射去找。

3. @Validated注解：参数检验

@Validated(value = {ChannelId.class, Keyword.class})，value中确认分组

@NotNull(groups = ChannelId.class)

private String channelId;

在实体类中，根据分组，决定要进行怎样的验证

4. <context:component-scan>标签

用于ApplicationContext.xml中，指定spring启动时，扫描某个包下的所有@Service,@Componet等标签，生成对应的bean

例：<context:component-scan base-package="com.cmb.o2o"/>

5. <aop:aspectj-autoproxy />标签

有一个proxy-target-class属性，默认为false，表示使用jdk动态代理织入增强，当配为<aop:aspectj-autoproxy poxy-target-class="true"/>时，表示使用CGLib动态代理技术织入增强。

6. CGLib动态代理

Spring AOP使用CGLib生成要代理类的子类，Cglib包的底层是通过使用一个小而块的字节码处理框架ASM来转换字节码并生成新的类.不鼓励直接使用ASM,因为它要求你必须对JVM内部结构包括class文件的格式和指令集都很熟悉。

7. Spring Boot初始化，Initialize函数

检查是否是web环境；

通过读取INF/spring.factories文件，使用classloader和反射的方法，创建相应的Initalizer实例以及listener实例，并加入SpringApplication类的容器中；

通过反射，检查main方法所在的类，记录下来。

8. Spring Boot初始化，run函数

1. StopWatch工具类，开始计时；

2. 设置系统java.awt.headless模式，对于没有外部设备的服务，性能会好一些；

3. 获取监听器，SpringApplicationRunListeners，调用started方法，启动所有监听，并且调用一部分监听器的onApplicationEvent方法，这里发起的是ApplicationStartedEvent。

4. 获取命令行的输入参数，生成ApplicationArguments实例，存放args

5. 生成WebEnvironment，发送监听器事件，发起的是ApplicationEnvironmentPreparedEvent

6. 打印Banner，根据Mode，分为打印到控制台还是log文件中

7. 创建ApplicationContext，web环境下，创建一个空的

AnnotationConfigEmbeddedWebApplicationContext实例

8. 创建FailureAnalyzers，不同的子类检查不同的内容，可以扩展自己的FailureAnalyzer，一旦启动过程进入catch，被注册的Analyzer实例的analyze方法就会被触发执行

9. 根据之前实例化的各种实例，构建一个完整的ApplicationContext：

prepareContext(context, environment, listeners, applicationArguments, printedBanner);

**有待仔细看**

9. Tomcat和Servlet的关系

Servlet的作用：封装了底层的Http操作，比如如何取出Http header，如何取出请求参数等

Tomcat的作用：封装了监听端口，管理Servlet，转发请求，管理HTTP连接池参数等功能，等于说Servlet是处理Http请求，而Tomcat负责从外部接受Http请求，以及返回请求结果。

10. webApplicationContext 与servletContext

ApplicationContext是Spring的核心，负责存放各种Spring管理的Bean

ServletContext是tomcat生成的，一个web应用中，所有的Servlet共享一个ServletContext，Servlet之间可以通过ServletContext通讯

ApplicationContext在准备完毕后，会存入ServletContext中。

11. 单例与多例嵌套关系

单例中包含多例，因为外层是单例，所以他的多例成员，永远都是同一个对象

在多例中包含单例，每次调用都会产生新的多例，但是单例永远是同一个对象

多例包含多例，每次调用两者都是新的实例

12. 使用XML文件，将Bean注入Spring容器中

标签：<bean>

Id：这个Bean的别名

class：Bean的类型

init-method：在Bean初始化完成后，执行的方法

property-value：成员名称-注入值

ref：已经生成的Bean的id

constructor-arg：使用构造器进行Bean的创建，通过该标签指定参数的值

Spring创建Bean时，并不是最初就根据依赖关系，决定所有Bean的创建顺序，而是一边读取XML文件一边创建，需要依赖时，再去创建依赖的对象。

13. Spring xml中使用其他配置文件中的Bean

<import resource=””>

Classpath代表classes文件夹。

14. @Autowired

Spring提供的自动注入的注解，自动去容器里寻找对应的Bean，必须保证只能找到一个对应的Bean，不然会抛出BeanCreationException，如果不确定是否有对应的Bean，可以设置required=false

例如 @Autowired(required=false)

@Autowired使用注入目标的类型，去Spring容器中搜索注入的Bean，当指定的类型有多个Bean时，会抛出异常说有多个候选项；为了解决这个问题，可以使用@Qualifer注解，指定Bean的id，确定唯一的Bean。

例如：@Autowired @Qualifer(value=”name”)

15. Spring解决循环依赖的问题

循环依赖：三个由Spring管理的Bean，ABC，A中有B，B中有C，C中有A，构成循环依赖

三种情况：

1. ABC都使用构造器构造，构造A时，发现需要B，将A放入缓存池，接着构造B，如此顺序，当构造C的时候，发现C依赖的A已经在缓存池中，即认为有循环依赖的问题，抛出异常：BeanCurrentlyInCreationException

2. ABC都是用set方式进行注入，使用setter方法注入时，会先创建Bean实例，再设置对象属性，所以首先创建A,B,C三个空的实例，在进行调用set方法设置成员的值，因此不会报错

3.ABC使用prototype的作用域，多例的创建过程都如同1中，不会提前暴露创建中的Bean，所以会抛出异常BeanCurrentlyInCreationException