# Spring学习笔记

## Spring注解

Spring容器负责对象的创建和释放。通过反射扫描对象所属的类是否含有注解，并采取预定义的注解所需做的操作。

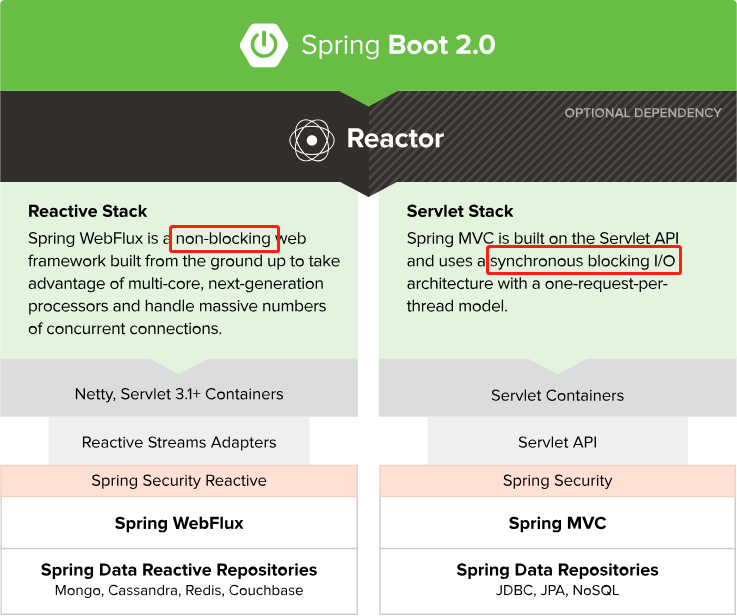
## 容器

容器即字面意思，是存放同一类对象的容器。这边的tomcat即是servlet容器。Spring的DI就是内部维护了Bean容器。

## Serializable序列化

## EventListener事件监听器

## Spring reactor



这里的Reactive和非Reactive的区别就是处理请求上的区别，是采用BIO还是NIO。

## 依赖注入（Dependency Injection）方式

1. 构造器注入
2. 接口注入

## 自动配置

通过@Component和 @ComponentScan 自动创建Bean, 通过@Autowired 完成Bean的自动装配。

可以同时使用不同的配置方式，只要功能健全即可。

## @Bean

添加了Bean注解的方法，Spring将会拦截所有对它的调用，并确保直接返回该方法创建的bean，而不是每次都对他进行实际的调用。

默认情况下，Spring中的bean都是单例的。

## Java不允许在同一个条目上重复出现相同类型的多个注解

## 硬编码和运行时注入

@Value注解，SpEL

Spring的properties在Environment对象中

## 切面AOP

需要知道，Spring使用了AspectJ的功能，单只是其子集。

## 切点表达式

## AspectJ切面

Spring bean由Spring容器初始化，但是AspectJ切面是由AspectJ在运行期创建的。即aspect切面对象不是由spring创建的。因此，等到Spring有机会为切面注入依赖时，切面已经被实例化了。需要使用Aspects.aspectof方法显示地创建Bean对象。然后spring会负责把相应的依赖进行注入。AspectOf方法会返回单例的切面对象。

**ATTENTION:**

Spring in action 4里是这么说，但实际测试下来，可以将切面设置为Component，也可以进行依赖注入，并不需要通过Aspects.aspectOf方法显示创建Bean。这样的话，反而会报错找不到切面的aspectOf方法，网上是说需要插件完成织入，但该插件又无法下载了 :(

总之，同时使用@Aspect注解和@Component注解完成相关的功能即可。

相关依赖

<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->  
<dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjrt</artifactId>  
 <version>1.9.5</version>  
</dependency>  
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->  
<dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>1.9.5</version>  
 <scope>runtime</scope>  
</dependency>

## Controller

Controller返回一个视图名，或者返回如"redirect:/spitter/" + spitter.getUsername();的字符串，视图解析器不会将其识别为视图名，而知道这个一个重定向。类型的还有forward。