Spring Framework注解编程的注解发展历史

一、Spring Framework 1.x 注解驱动启蒙时代

- 从Spring Framework 1.2.0版本开始，开始支持Annotation，虽然框架层面均已经支持**@managedResource**和**@Transactional**等注解，但主要还是以XML配置为准

二、Spring Framework 2.x 注解驱动过渡时期

- Spring Framework 2.0在Anonotation支持方面添加了新的成员，**@Required、数据相关的@Repository及AOP相关的@Aspect等**，但同时提升了XML配置能力，即“可扩展的XML编写”。

三、Spring Framework 2.5 新的骨架式Anonotation

- @Service
- @Controller，@RequestMapping以及@ModelAttribut
- @Resource注入
- @PostConstruct替代
- @PreDestory替代
  尽管此时提供了不少的注解，但编程的手段却不多，主要原因在于框架层面仍未“直接”提供驱动注解，仍需要XML驱动。

四、Spring Framework 3.x 注解驱动黄金时代

- Spring Framework 3.x是一个里程牌式的时代，在3.0时代中除了提升Spring模式注解“派生”的层次性，并替换XML配置方式，引入**配置类注解@Configuration**，该注解是内建的**@Component**的“派生”注解。同时使用**@ImportResource和@Import**。@ImportResource负责导入遗留的XML配置文件，@Import允许导入一个或多个类作为SpringBean
- 3.1引入注解**@ComponentScan**，替换XML的context:component-scan，距离全面注解驱动更近一步

五、Spring Framework 4.x 注解驱动完善时代

- 3.1开始提供@Profil提供了配置化的条件组装，但这一方面仍为比较单一，4.0之后，引入**条件注解@Conditional**。通过自定义Condition实现配合，弥补了之前版本条件化装配的短板，4.0开始Profile反过来通过@Conditional实现
- 4.2开始新增**事件监听器注解@EventListener**，作为ApplicationListener接口编程的第二选择。**@AliasFor解除注解派生的时候冲突限制。**
- 在浏览器跨域资源访问方面，引入**@CrossOrigin**，作为CorsRegistry替换注解方案。

六、Spring Framework 5.x 注解驱动成熟时代

- SpringFramework5.0作为Spring Boot2.0的底层，注解驱动的性能提升方面不是很明显。**在Spring Boot应用场景中，大量使用@ComponentScan扫描，导致Spring模式的注解解析时间耗时增大**，因此，5.0时代引入**@Indexed**，为Spring模式注解添加索引。