## 1.什么是Springboot？

SpringBoot是一个快速开发的框架，能过快速帮我们整合常用的第三方框架;

## 2.Springboot如何整合第三方框架的？

基于Maven的继承方式,完全采用注解化(使用注解的方式开启SpringMVC),简化XML配置，内置HTTP服务器（Tomcat，Jetty），最终以java应用程序进行执行；无需发布打包到服务器,直接通过Java命令即可执行(java -jar)

## 3.为什么使用Springboot？

**传统项目：**

a.整合SSH或SSM，考虑 配置文件，jar包冲突问题,整合起来繁琐;

打包方式：打成一个war包放入到tomacat的webapps目录下进行执行;

**Springboot项目：**

a.Springboot 是一个快速开发框架，能够帮助我们快速整合第三方常用框架（Maven依赖关系（Maven的继承）），完全使用注解化，简化xml文件配置(在spring3.0之后采用注解方式启动Springmvc),内置嵌入了Http服务器（Tomcat,Jetty)，默认嵌入了Tomacat服务器;最终以java应用程序进行执行;使用Java创建Tomacat;Springboot项目没有web.xml;

## 4.Springboot与SpringMVC的关系？

a.Springboot web组件集成了SpringMvc框架;但是Springboot启动SpringMVc是没有传统的配置文件;

b.SpringBoot如何启动SpringMvc？

在SpringMVC3.0支持注解方式启动;

## 5. Springboot与SpringCould的关系？

a.Springboot是一个快速开发框架，可以快速帮助我们整理常用第三方框架；

完全采用注解化，简化xml文件配置，最终以java应用程序进行执行;

b.SpringCould是一套目前完整微服务解决框架;

注册中心，客户端调用工具，服务治理（负载均衡，断路器，分布式配置中心，网关，服务链路等）;

**c.关系：**

**Springboot是 微服务框架？**

SpringBoot+SpringCould 实现微服务开发;

SpringBoot实现快速开发;

微服务通讯技术http+json(restful)轻量级Springboot Web组件默认集成了SpringMVC，SpringCould依赖于Springboot实现微服务, 使用SpringMVC编写微服务接口;

## 6.springboot2.X自动装配原理（HttpEncodingAutoConfiguration为例）

@Configuration(proxyBeanMethods = **false**)

@EnableConfigurationProperties(HttpProperties.**class**)

@ConditionalOnWebApplication(type = ConditionalOnWebApplication.Type.***SERVLET***)

@ConditionalOnClass(CharacterEncodingFilter.**class**)

@ConditionalOnProperty(prefix = "spring.http.encoding", value = "enabled", matchIfMissing = **true**)

**public** **class** HttpEncodingAutoConfiguration {

}

@ConfigurationProperties(prefix = "spring.http")

**public** **class** HttpProperties {

}

1.@Configuration：标识此类为配置类,将此类纳入SpringIOC容器;

2.@EnableConfigurationProperties(HttpProperties.**class**):

通过HttpProperties将默认编码设置为UTF-8（自动装配为UTF-8）

3.修改默认编码

通过HttpProperties中的prefix+属性名（charset）进行修改[在配置文件中yml/properties, 该注解提供了默认的utf-8，并提供了prefix+属性名的方式来修改编码;在配置文件中配置; spring.http.encoding.charset=ISO-8859-1

4. @ConditionalOnWebApplication(type = ConditionalOnWebApplication.Type.***SERVLET***)

仅在当前应用是 Servlet Web 应用时才生效@ConditionalOnClass(CharacterEncodingFilter.**class**)

仅在类 CharacterEncodingFilter 存在于 classpath 上时才生效@ConditionalOnProperty(prefix = "spring.http.encoding", value = "enabled", matchIfMissing = **true**)

仅在配置项 spring.http.encoding.enabled 不存在或者设置为 true 时才生效

Conditional条件;on当...的时候;WebApplication web应用;

value:等价于name;

prefix:前缀;

matchIfMissing:如果没有匹配则满足;

Conditional其他注解:

@ConditionalOnMissingBean:容器中不存在指定Bean时生效;

@ConditionalOnBean:容器中存在指定Bean时生效;

@ConditionalOnMissingClass:容器中不存在指定类时生效;

@ConditionalOnClass:容器中存在指定类是生效;

@ConditionalOnNotWebApplication:当前不是web环境时生效;

@ConditionalOnWebApplication:当前是web环境时生效;

@ConditionalOnProperty:系统中指定的属性是否有指定的值;

@ConditionalOnJava:系统中java版本是否符合要求;

@ConditionalOnExpression:满足指定SpEl表达式;

@ConditionalOnSingleCandidate:容器中只有一个指定的Bean，或者这个Bean是首选Bean;

@ConditionalOnResource:类路径下存在指定的资源文件生效;

@ConditionalOnJndi:JNDI存在指定项;

总结：

1.每个XxAutoConfiguration都有很多@ConditionalOnXx,当这些条件满足是,则此配置自动生效; 但是可以手工修改自动装配XxxProperties中的prefix+属性名=value;

2.全局配置文件中的key,来源于某个XxxProperties中的prefix+属性名=value;