**一、注解(annotations)列表**   
@SpringBootApplication：包含了@ComponentScan、@Configuration和@EnableAutoConfiguration注解。其中@ComponentScan让[spring](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "_blank) Boot扫描到Configuration类并把它加入到程序上下文。

@Configuration 等同于spring的XML配置文件；使用[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "_blank)代码可以检查类型安全。

@EnableAutoConfiguration 自动配置。

@ComponentScan 组件扫描，可自动发现和装配一些Bean。

@Component可配合CommandLineRunner使用，在程序启动后执行一些基础任务。

@RestController注解是@Controller和@ResponseBody的合集,表示这是个控制器bean,并且是将函数的返回值直 接填入HTTP响应体中,是REST风格的控制器。

@Autowired自动导入。

@PathVariable获取参数。

@JsonBackReference解决嵌套外链问题。

@RepositoryRestResourcepublic配合spring-boot-starter-data-rest使用。

**二、注解(annotations)详解**

@SpringBootApplication：申明让spring boot自动给程序进行必要的配置，这个配置等同于：@Configuration ，@EnableAutoConfiguration 和 @ComponentScan 三个配置。

[复制代码](javascript:void(0);)

package com.example.myproject;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication // same as @Configuration @EnableAutoConfiguration @ComponentScan

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

[复制代码](javascript:void(0);)

@ResponseBody：表示该方法的返回结果直接写入HTTP response body中，一般在异步获取数据时使用，用于构建RESTful的api。在使用@RequestMapping后，返回值通常解析为跳转路径，加上@responsebody后返回结果不会被解析为跳转路径，而是直接写入HTTP response body中。比如异步获取json数据，加上@responsebody后，会直接返回json数据。该注解一般会配合@RequestMapping一起使用。示例代码：

@RequestMapping(“/test”)

@ResponseBody

public String test(){

return”ok”;

}

@Controller：用于定义控制器类，在spring 项目中由控制器负责将用户发来的URL请求转发到对应的服务接口（service层），一般这个注解在类中，通常方法需要配合注解@RequestMapping。示例代码：

[复制代码](javascript:void(0);)

@Controller

@RequestMapping(“/demoInfo”)

publicclass DemoController {

@Autowired

private DemoInfoService demoInfoService;

@RequestMapping("/hello")

public String hello(Map<String,Object> map){

System.out.println("DemoController.hello()");

map.put("hello","from TemplateController.helloHtml");

//会使用hello.html或者hello.ftl模板进行渲染显示.

return"/hello";

}

}

[复制代码](javascript:void(0);)

@RestController：用于标注控制层组件(如struts中的action)，@ResponseBody和@Controller的合集。示例代码：

[复制代码](javascript:void(0);)

package com.kfit.demo.web;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping(“/demoInfo2”)

publicclass DemoController2 {

@RequestMapping("/test")

public String test(){

return"ok";

}

}

[复制代码](javascript:void(0);)

@RequestMapping：提供路由信息，负责URL到Controller中的具体函数的映射。

@EnableAutoConfiguration：Spring Boot自动配置（auto-configuration）：尝试根据你添加的jar依赖自动配置你的Spring应用。例如，如果你的classpath下存在HSQLDB，并且你没有手动配置任何[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)连接beans，那么我们将自动配置一个内存型（in-memory）数据库”。你可以将@EnableAutoConfiguration或者@SpringBootApplication注解添加到一个@Configuration类上来选择自动配置。如果发现应用了你不想要的特定自动配置类，你可以使用@EnableAutoConfiguration注解的排除属性来禁用它们。

@ComponentScan：表示将该类自动发现扫描组件。个人理解相当于，如果扫描到有@Component、@Controller、@Service等这些注解的类，并注册为Bean，可以自动收集所有的Spring组件，包括@Configuration类。我们经常使用@ComponentScan注解搜索beans，并结合@Autowired注解导入。可以自动收集所有的Spring组件，包括@Configuration类。我们经常使用@ComponentScan注解搜索beans，并结合@Autowired注解导入。如果没有配置的话，Spring Boot会扫描启动类所在包下以及子包下的使用了@Service,@Repository等注解的类。

@Configuration：相当于传统的xml配置文件，如果有些第三方库需要用到xml文件，建议仍然通过@Configuration类作为项目的配置主类——可以使用@ImportResource注解加载xml配置文件。

@Import：用来导入其他配置类。

@ImportResource：用来加载xml配置文件。

@Autowired：自动导入依赖的bean

@Service：一般用于修饰service层的组件

@Repository：使用@Repository注解可以确保DAO或者repositories提供异常转译，这个注解修饰的DAO或者repositories类会被ComponetScan发现并配置，同时也不需要为它们提供XML配置项。

@Bean：用@Bean标注方法等价于XML中配置的bean。

@Value：注入Spring boot application.properties配置的属性的值。示例代码：

@Value(value = “#{message}”)

private String message;

@Inject：等价于默认的@Autowired，只是没有required属性；

@Component：泛指组件，当组件不好归类的时候，我们可以使用这个注解进行标注。

@Bean:相当于XML中的,放在方法的上面，而不是类，意思是产生一个bean,并交给spring管理。

@AutoWired：自动导入依赖的bean。byType方式。把配置好的Bean拿来用，完成属性、方法的组装，它可以对类成员变量、方法及构造函数进行标注，完成自动装配的工作。当加上（required=false）时，就算找不到bean也不报错。

@Qualifier：当有多个同一类型的Bean时，可以用@Qualifier(“name”)来指定。与@Autowired配合使用。@Qualifier限定描述符除了能根据名字进行注入，但能进行更细粒度的控制如何选择候选者，具体使用方式如下：

@Autowired

@Qualifier(value = “demoInfoService”)

private DemoInfoService demoInfoService;

@Resource(name=”name”,type=”type”)：没有括号内内容的话，默认byName。与@Autowired干类似的事。

**三、JPA注解**

@Entity：@Table(name=”“)：表明这是一个实体类。一般用于jpa这两个注解一般一块使用，但是如果表名和实体类名相同的话，@Table可以省略

@MappedSuperClass:用在确定是父类的entity上。父类的属性子类可以继承。

@NoRepositoryBean:一般用作父类的repository，有这个注解，spring不会去实例化该repository。

@Column：如果字段名与列名相同，则可以省略。

@Id：表示该属性为主键。

@GeneratedValue(strategy = GenerationType.SEQUENCE,generator = “repair\_seq”)：表示主键生成策略是sequence（可以为Auto、IDENTITY、native等，Auto表示可在多个数据库间切换），指定sequence的名字是repair\_seq。

@SequenceGeneretor(name = “repair\_seq”, sequenceName = “seq\_repair”, allocationSize = 1)：name为sequence的名称，以便使用，sequenceName为数据库的sequence名称，两个名称可以一致。

@Transient：表示该属性并非一个到数据库表的字段的映射,ORM框架将忽略该属性。如果一个属性并非数据库表的字段映射,就务必将其标示为@Transient,否则,ORM框架默认其注解为@Basic。@Basic(fetch=FetchType.LAZY)：标记可以指定实体属性的加载方式

@JsonIgnore：作用是json序列化时将[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)bean中的一些属性忽略掉,序列化和反序列化都受影响。

@JoinColumn（name=”loginId”）:一对一：本表中指向另一个表的外键。一对多：另一个表指向本表的外键。

@OneToOne、@OneToMany、@ManyToOne：对应[hibernate](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "_blank)配置文件中的一对一，一对多，多对一。

**四、springMVC相关注解**

@RequestMapping：@RequestMapping(“/path”)表示该控制器处理所有“/path”的UR L请求。RequestMapping是一个用来处理请求地址映射的注解，可用于类或方法上。   
用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。该注解有六个属性：   
params:指定request中必须包含某些参数值是，才让该方法处理。   
headers:指定request中必须包含某些指定的header值，才能让该方法处理请求。   
value:指定请求的实际地址，指定的地址可以是URI Template 模式   
method:指定请求的method类型， GET、POST、PUT、DELETE等   
consumes:指定处理请求的提交内容类型（Content-Type），如application/json,text/html;   
produces:指定返回的内容类型，仅当request请求头中的(Accept)类型中包含该指定类型才返回

@RequestParam：用在方法的参数前面。   
@RequestParam   
String a =request.getParameter(“a”)。

@PathVariable:路径变量。如

RequestMapping(“user/get/mac/{macAddress}”)

public String getByMacAddress(@PathVariable String macAddress){

//do something;

}

参数与大括号里的名字一样要相同。

**五、全局异常处理**

@ControllerAdvice：包含@Component。可以被扫描到。统一处理异常。

@ExceptionHandler（Exception.class）：用在方法上面表示遇到这个异常就执行以下方法。