# Springboot笔记

## 一、注解

1、@ModelAttribute：等同于通过映射将后台数据传递给前端（ModelAndView.addObject(“xxx”,xxx)）

2、@WebMvcTest：测试mvc自动生成Mockmvc

@MockBean：用在引用的被测试类上，实现自动装载

3、@GetMapping:处理GET请求

@PostMapping：处理POST请求

@PutMapping：处理PUT请求

@DeleteMapping处理DELETE请求

4、@ConfigurationProperties:加载properties文件内的配置，通过prefix指定前缀

5、@Configuration：标注在类上，相当于把该类作为spring的xml配置文件中的<beans>结合@Bean一起使用

6、@Bean：标注在方法上(返回某个实例的方法)，等价于spring的xml配置文件中的<bean>

7、@Scope：默认是单例模式（singleton），singleton单例模式,全局有且仅有一个实例，prototype原型模式，每次获取Bean的时候会有一个新的实例，request表示该针对每一次HTTP请求都会产生一个新的bean

8、@ComponentScan等价于<context:component-scan base-package="com.dxz.demo"/>

9、@Import: 引入带有@Configuration的java类

10、@ControllerAdvice：controller的aop的切面，作用在所有控制器之上，可以结合@ExceptionHandler、@InitBinder、@ModelAttribute一起使用，使它们全局生效。

11、Repository:声明此类是一个数据库或者其他nosql访问类（dao层）

12、@PostConstruct:标注在方法上，当bean被容器初始化后，就调用该方法。

13、@PreDestory:容器被销毁之前，调用被注解的方法。

14、@Qualifier:根据名字指定哪个实现类才是我们需要的。

15、@ExceptionHandler(RuntimeException.**class**)：标注在统一异常处理方法上，结合@ControllerAdvice使用。

16、@InitBinder:注册一个属性编辑器，只对当前的Controller生效，如果要全局生效，需要使用在@ControllerAdvice注解的类下。

17、@Aspect:声明这是一个切面类。

18、@Around:环绕通知，声明一个表达式，比如1."@within(org.springframework.stereotype.Controller)：所有controller中的方法。

2.execution(public \* \*(…))：所有public方法。

3.execution(public \* set\*(…))：所有set开头方法。

4.execution(public \* set\*(…))：所有set开头的public方法。

5.execution(public com.xyz.\* set\*(…))：所有com.xyz包下set开头的public方法。

6.target(com.xyz.ComService):所有实现了ComService接口的类的方法。

7.@target(org. springframework.transaction.annotation.Transactional)：所有@ Transactional注解的方法

19、@ConditionalOnBean:在当前上下文中存在某个对象时，才会实例化一个Bean

20、@ConditionalOnMissingBean:在当前上下文不存在某个对象时，才会实例化一个bean

21、

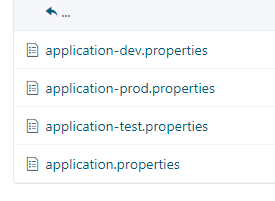
### 二、mvc的测试

*/\*\*  
 \* 1、mockMvc.perform执行一个请求。  
 \* 2、MockMvcRequestBuilders.get("XXX")构造一个请求。  
 \* 3、ResultActions.param添加请求传值  
 \* 4、ResultActions.accept(MediaType.TEXT\_HTML\_VALUE))设置返回类型  
 \* 5、ResultActions.andExpect添加执行完成后的断言。  
 \* 6、ResultActions.andDo添加一个结果处理器，表示要对结果做点什么事情  
 \* 比如此处使用MockMvcResultHandlers.print()输出整个响应结果信息。   
 \* 5、ResultActions.andReturn表示执行完成后返回相应的结果。  
 \*/*

@Autowired  
**private** WebApplicationContext **webApplicationContext**;  
**private** MockMvc **mockMvc**;  
@Before  
**public void** setUp() **throws** Exception {  
 **mockMvc** = MockMvcBuilders.*webAppContextSetup*(**webApplicationContext**).build();  
}  
@Test**public void** contextLoads() **throws** Exception {  
 **mockMvc**.perform(MockMvcRequestBuilders.*get*(**"/allUserCount"**)  
 .accept(MediaType.***APPLICATION\_JSON***))  
 *//.andExpect(MockMvcResultMatchers.status().isOk())* .andDo(MockMvcResultHandlers.*print*())  
 .andReturn();  
 *//.andExpect(MockMvcResultMatchers.content().string(""))* ;  
  
}

## 三、配置文件（常用）

1、环境端口号配置：



2、端口号：server.port=

3、Tomcat的相关配置：

Tomcat访问日志：serer.tomcat.accesslog.enabled=false

4、日志配置：loggin.level.root=info(root开头的类)

Loggin.level.org=warn（org开头的类）