Springboot

微服务 每一个部分都是一个简单的功能单元

Spring cloud 进行分布的设置 等spring的分布式架构

Springboot 的简单的配置 （父项目）

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.1.RELEASE</version>

</parent>

就是所有的 springboot的管理 （所有依赖版本）

如果缺少的话，自己来添加相应的版本号

场景启动器 web相应需要的模块 （start就是相应依赖的 启动器）

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

主程序类

@SpringBootApplication 其实是一个组合注解

标注在某个类上 ，说明是springboot 的主配置类

@Target(ElementType.TYPE)

@Retention(RetentionPolicy.RUNTIME)

@Documented

@Inherited

@SpringBootConfiguration

@EnableAutoConfiguration

@ComponentScan(excludeFilters = {

@Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),

@Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class)

1 @SpringBootConfiguration 表示这是一个springboot的配置类

@Configuration 配置类 其实就是一个容器中的组件

2@EnableAutoConfiguration 开启自动配置类

@AutoConfigurationPackage **自动配置包**

@Import(AutoConfigurationImportSelector.class)

@Import(AutoConfigurationPackages.Registrar.class)

将主配置类 （ @SpringBootApplication 标注的类） 的所有子包里面的组件扫描到spring容器中

@Import(AutoConfigurationImportSelector.class) 会给容器导入需要场景的配置类

3 使用spring Initializer 快速创建spring boot 项目

选择需要的模块，联网自动创建springboot项目

1) 主程序已经写好只需要写逻辑部分

2) resource 文件的目录结构

Static 就是所有的静态资源

Templates （保存所有的模板页面） springboot 使用的是嵌入式的tomcat 不支持jsp 可以使用 freemarker ，thymeleaf

Application.properities 应用的配置文件 修改相应的配置相当方便 （比如配置tomcat的端口）

4 springboot的配置文件

A 使用的全局配置文件 （名称是固定的）

Application.properties

Application.yml

作用修改springboot 默认配置

Yml配置问价 ( YAML Ain’t Markup Language) 以数据为中心 比xml和json更适合配置

具体的配置实例

server:

address: 1234

和之前的xml完全不同

Yaml 的基本语法

1 空格缩进就是表示不同的层间关系

2 值的写法

A 字面量 普通的值 （数字，字符串，布尔）

字符串默认不加单引号或者双引号

双引号不会转义字符串，特殊字符会作为本身的意思

Name : “zhangsan \n lis” 输出 zhangsan 换行 lis

单引号 会转移特殊字符 ，特殊字符最终只是一个普通的字符串数据

Name : “zhangsan \n lis” 输出 zhangsan \n lis

B 对象 map(属性 键值对) 注意缩进

Friends :

lastName :zhangsan

age :20

行内写法

Frineds :{ lastname：zhangsan ,age:20}

C 数组 （List set）

Pets:

* Cat
* Dog

行内写法

Pets: [ cat,dog]

测试配置文件 （需要进行绑定参数）

@ConfigurationProperties(prefix="person")

可能需要导入相应的处理器

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

如果是传入相应的值的话 两种配置方式的数值传入都是可以的 （只是格式不一样）

@ConfigurationProperties 功能可以批量的进入注入 （支持松散绑定）

补充 SpEl 的表达式 一般是在配置文件中注入或者是在注解中使用就是value调用的一种方式

 @Value("#{systemProperties['user.language']}")

或者是

1. **<bean** id="MyMessage" class="cn.spy.spel.injection.MyMessage"**>**
2. **<property** name="message" value="#{systemProperties['user.language']}"**></property>**
3. **</bean>**

松散绑定 字母之间有-符号

JSR303 数据校检

|  |  |  |  |
| --- | --- | --- | --- |
|  | @ConfigurationProperties | @value |  |
| 功能 | 批量注入配置文件属性值 | 一个个指定 |  |
| 松散绑定 | 支持 | 不支持 |  |
| SPEL | 不支持 | 支持 |  |

|  |  |  |
| --- | --- | --- |
| JSR303 | 支持 | 不支持 |
| 复杂类型分装 | 支持 | 不支持 |

如果只是在某个业务下直接从配置文件中取出值 就使用@value

如果专门写一个 javabean 来和配置文件进行映射我们就直接使用@ConfigurationProperties

校检部分实现 （对Javabean实现校检）

@ConfigurationProperties(prefix="person")

@Validated

public class person {

@NotNull

private String lastname;

@ConfigurationProperties(prefix="person") 默认使用的全局配置

@PropertySource 来加载制定的配置文件 @PropertySource("classpath:config.properties")

通过@ImportResource实现xml配置的装载

Spring boot 没有 spring的配置文件 自己编写的也不能识别。想让spring的配置文件生效的话 加载进来@ImportResource标注一个配置文件

补充junit 测试容器中是否含有bean 的方法

首先编写 bean.xml 文件 然后再主方法上添加

在相应的test文件中添加

@Autowired

ApplicationContext ioc;

@Test

public void testHelloService() {

ioc.containBean(name:"helloService");

}

最后的添加是

@ImportResource(locations="classpath:beanx.xml")

Springboot 推荐添加组件的方式 推荐使用注解的方式

1 配置类 相当于spring的配置文件 并且不需要@ImportResource 添加配置

2 使用Bean给容器添加组件

@Configuration

public class MyAppConfig {

@Bean

public HelloService helloService() {

System.out.println("laalla");

return new HelloService();

}

配置文件的占位符

RandomValuePropertySource ： 配置文件中 可以使用随机数

${random.value} ${random.int}

${random.uuid}

配置文件占位符如果没有的话 可以是用 制定默认值

person.lastname=zhangsan ${random.uuid}

person.age=${random.int}