# 配置文件

SpingBoot配置文件在src/resources目录下，或者src/resources/config目录。

application.properties或者application.yml。

更改配置文件名：

1. 在启动项目时，启动参数：--spring.config.name=newName，只需要文件名，扩展名可以省略。
2. 在启动项目时，启动参数：--spring.config.location=classpath:path。可以指定多个，使用逗号隔开。

获取配置文件的值：

|  |
| --- |
| @Component public class Tomcat {  @Value("${tomcat.ip}")  private String ip ;   @Value("${tomcat.port}")  private int port ;   @Value("${tomcat.defaultPort:8080}")  private int defaultPort ;   @Override  public String toString(){  return "Tomcat{" + "ip='" + ip + '\'' + ", port=" + port + '}';  } } |

Application.properties配置文件：

tomcat.ip=192.168.216.120  
tomcat.port=80

使用@PropertyResource和@PropertyResources注解加载额外的配置文件。

|  |
| --- |
| @PropertySource("classpath:jdbc.properties") @Component public class SqlConfig {  @Value("${jdbc.user}")  private String user ;   @Value("${jdbc.password}")  private String password ;   @Value("${jdbc.url}")  private String url ;   @Value("${jdbc.driver}")  private String driver ;    @Override  public String toString(){  return "SqlConfig{" + "user='" + user + '\'' + ", password='" + password + '\'' + ", url='" + url + '\'' + ", driver='" + driver + '\'' + '}';  } } |

Jdbc.properties:

jdbc.user=root  
jdbc.password=lsh159  
jdbc.url=jdbc:mysql://localhost:3306/spring\_boot  
jdbc.driver=com.mysql.jdbc.Driver

使用@ConfigurationProperties注解加载默认的配置文件（application.properties）中的指定前缀的value。

|  |
| --- |
| @ConfigurationProperties(prefix = "jdbc") @Component public class SqlConfig {  private String user ;  private String password ;  private String url ;  private String driver ;  public String getUser(){  return user;  }  public void setUser(String user){  this.user = user;  }  public String getPassword(){  return password;  }  public void setPassword(String password){  this.password = password;  }  public String getUrl(){  return url;  }  public void setUrl(String url){  this.url = url;  }  public String getDriver(){  return driver;  }  public void setDriver(String driver){  this.driver = driver;  }  @Override  public String toString(){  return "SqlConfig{" + "user='" + user + '\'' + ", password='" + password + '\'' + ", url='" + url + '\'' + ", driver='" + driver + '\'' + '}';  } } |

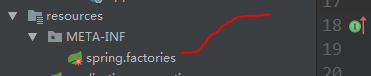
该注解支持数组和集合的注入：prefix.name[index]=value

实现接口org.springframework.boot.env.EnvironmentPostProcessor接口加载额外的配制文件。

第一步：实现接口。

|  |
| --- |
| @Component public class MyEnvironmentPostProcessor implements EnvironmentPostProcessor {  public void postProcessEnvironment(ConfigurableEnvironment environment, SpringApplication application)  {  try  {  InputStream input = this.getClass().getClassLoader().getResourceAsStream("jdbc.properties") ;  Properties sources = new Properties();  sources.load(input);  PropertiesPropertySource propertySource = new PropertiesPropertySource("my",sources) ;  environment.getPropertySources().addLast(propertySource);  }catch(IOException e){  e.printStackTrace();  }  } } |

第二步：配置META-INF/spring.factories文件。



org.springframework.boot.env.EnvironmentPostProcessor=code.lsh.inter.MyEnvironmentPostProcessor

在不同的环境中加载不同的配置文件：

如：application-dev.properties：开发时加载的配置文件

application-test.properties：测试时加载的配置文件

application-product.proterties：生产时加载的配置文件

方法一：手动指定。

SpringApplication application = new SpringApplication(Application.class);  
//加载不同的配置文件  
application.setAdditionalProfiles("dev");  
ConfigurableApplicationContext context = application.run(args) ;  
System.*out*.println(context.getEnvironment().getProperty("app.name"));  
context.close();

方法二：指定JVM启动参数：--spring.profiles.active=dev,test…..

可以激活多个配置文件，使用逗号（,）分开。

方法三：使用注解@Profile()，该注解也可以使用在类上。

@Bean("person")  
@Profile("dev")//在application-dev.properties文件激活时，才装配该Bean  
public Person createPerson()  
{  
 Person person = new Person();  
 person.setName("lisaihao");  
 person.setAge(20);  
 return person;  
}

# 自动配置

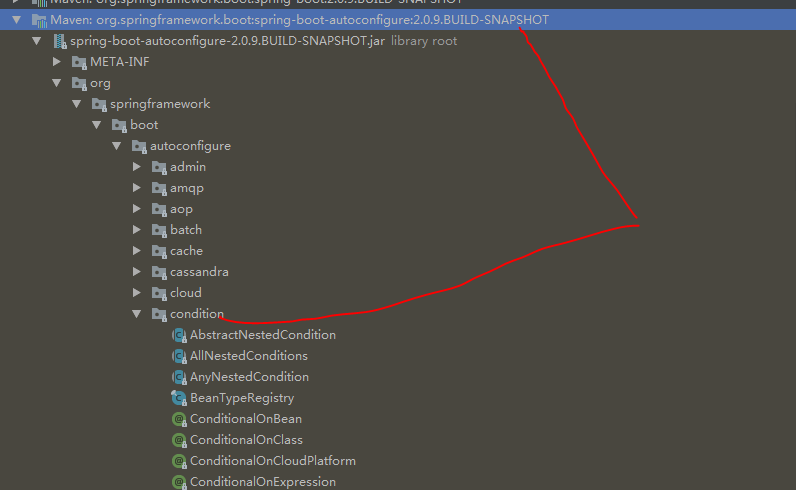
org.springframework.context.annotation.Condition该接口中是Spring提供的

org.springframework.context.annotation.Conditional注解，该注解是基于条件的注解，接受一个实现了Condition接口的class，只有实现类的matcher方法返回true时才会装配。该注解也可以使用到类上。

@Bean  
@Conditional(UTF8Condation.class)  
public UTF8Character createUTF()  
{  
 return new UTF8Character();  
}

SpringBoot为我们提供了许多自动配置注解：

所在的jar包：



@ConditionalOnBean ，@ ConditionalOnClass，@ConditionalOnWebApplication

@ConditionalOnResource，@ ConditionalOnProperty，

@Enable\*注解：

@EnableAutoConfiguration

@EnableConfigurationProperties

@EnableAsync

@Import注解：

ImportSelector接口：

ImportBeanDefinitionRegistrar接口

http://patorjk.com/