

SpringBoot配置方法

课程目标

- Application的配置及获取
- 自定义配置文件的配置及获取
- XML文件的配置及获取

application文件的配置

SpringBoot的配置

□ SpringBoot的配置使用全局的配置文件

➤ application.properties

➤ application.yml

□ 配置文件路径(自动加载):

➤ src/main/resource目录下

➤ 或类路径下的/config下。

□ 注：这两种配置文件的区别是格式不一致，可互相替换，以后的主流应该是以yml为主的。

application.properties配置

□ application.properties的配置以属性名=属性值得形式进行配置,注意行中(末尾)不要有空格,不要有空行.

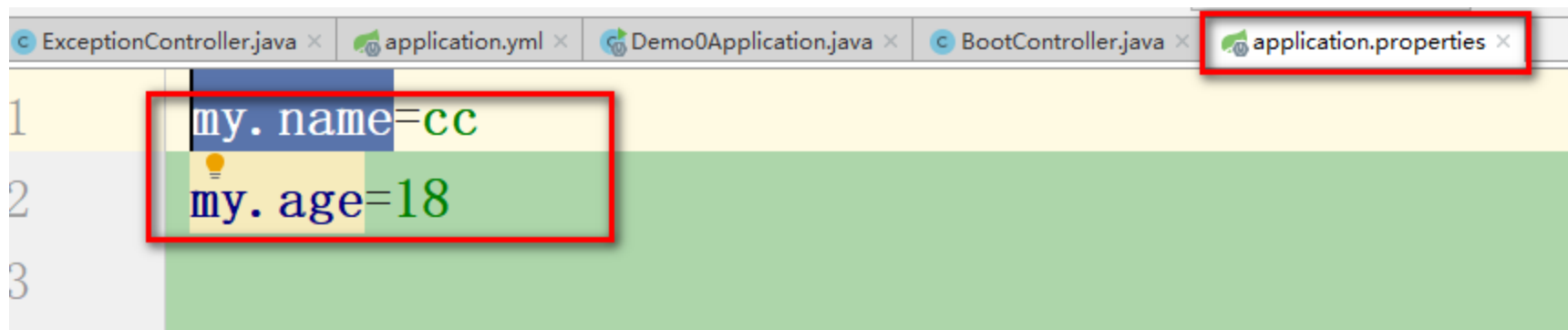
□ 实例:

```
spring.jpa.database=oracle  
spring.datasource.url=jdbc:oracle:thin:@127.0.0.1:1521:orcl  
spring.datasource.username=java05  
spring.datasource.password=1  
spring.datasource.driverClassName=oracle.jdbc.OracleDriver  
spring.datasource.max-active=20
```

获取主配置文件的配置项

- 当在application.properties配置文件中自定义配置项时,可以使用@Value(“\${}”)注解或定义类的方式获取自定义配置项的值.
- 注意使用@Value时,所在类必须被Spring容器管理,也就是必须被@Controller,@Service等注解.

□ 配置文件



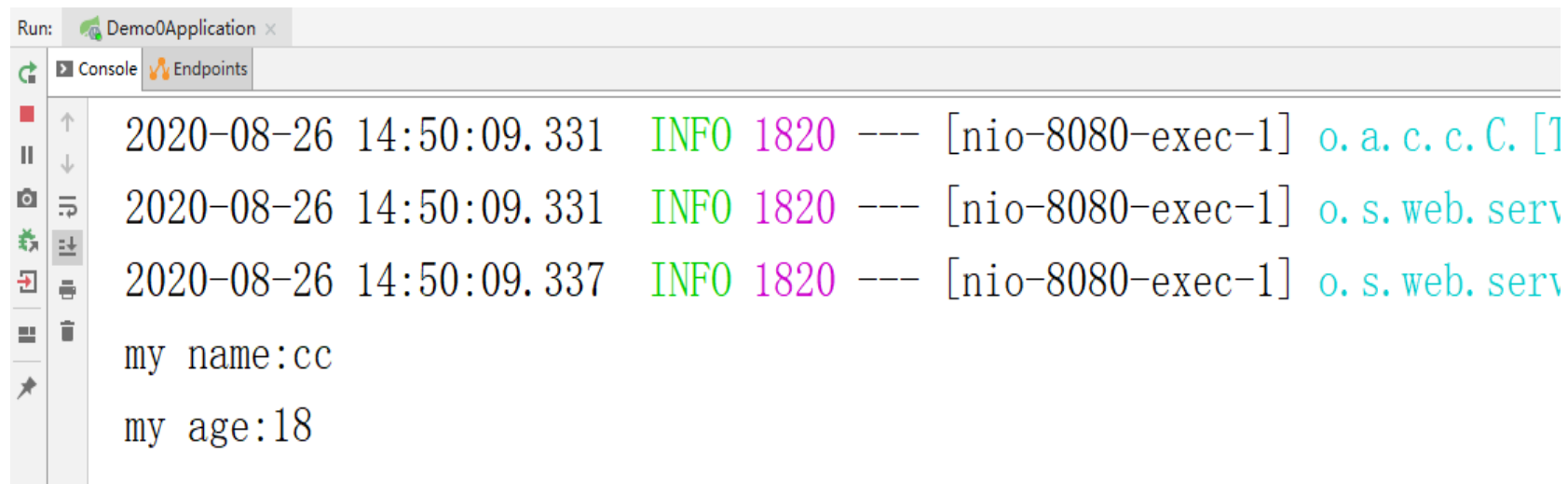
```
ExceptionHandler.java × application.yml × Demo0Application.java × BootController.java × application.properties ×  
1 my.name=cc  
2 my.age=18  
3
```

□Java代码

```
@RestController
public class HelloController {
    @Value("${my.name}")
    private String name;
    @Value("${my.age}")
    private int age;

    @RequestMapping("hello")
    public String hello() {
        System.out.println("my name:" + name);
        System.out.println("my age:" + age);
        return "HelloWorld!";
    }
}
```


□运行结果



The screenshot shows an IDE's console window for a project named 'Demo0Application'. The 'Console' tab is active, displaying three log messages with timestamps, log levels, thread names, and class names. The messages are:

- 2020-08-26 14:50:09.331 INFO 1820 --- [nio-8080-exec-1] o.a.c.c.C.[1
- 2020-08-26 14:50:09.331 INFO 1820 --- [nio-8080-exec-1] o.s.web.serv
- 2020-08-26 14:50:09.337 INFO 1820 --- [nio-8080-exec-1] o.s.web.serv

Below the log messages, the user has entered two lines of text:

- my name:cc
- my age:18

The IDE interface includes a toolbar on the left with icons for running, debugging, and other actions, and a tab bar at the top showing the current application.

□ 常见配置项可参考:<<示例application.properties
>>

□ 完整可参考:

➤ <https://docs.spring.io/spring-boot/docs/current/reference/html/appendix-application-properties.html>

application.yml配置

□在yml之前使用最多的配置文件形式是xml和properties文件。

- xml文件太过繁琐，看过的人都知道，想要新加一个配置节点的话还需要包含在<>标签里；
- 而properties配置文件没有了标签，不过当你的配置有很多层级的时候，写完之后你会发现会有大量重复的代码。

□而yaml/yaml文件结合了两者的优势，当你新增节点配置的时候，不需要标签，在写多层级配置的时候也不会产生重复代码。

□Yml书写时需要注意：

- 使用空格的缩进表示层级关系，空格数目不重要，只要是左对齐的一列数据，都是同一个层级的。
- 大小写敏感
- 缩进时不允许使用Tab键，只允许使用空格。

application.yml配置

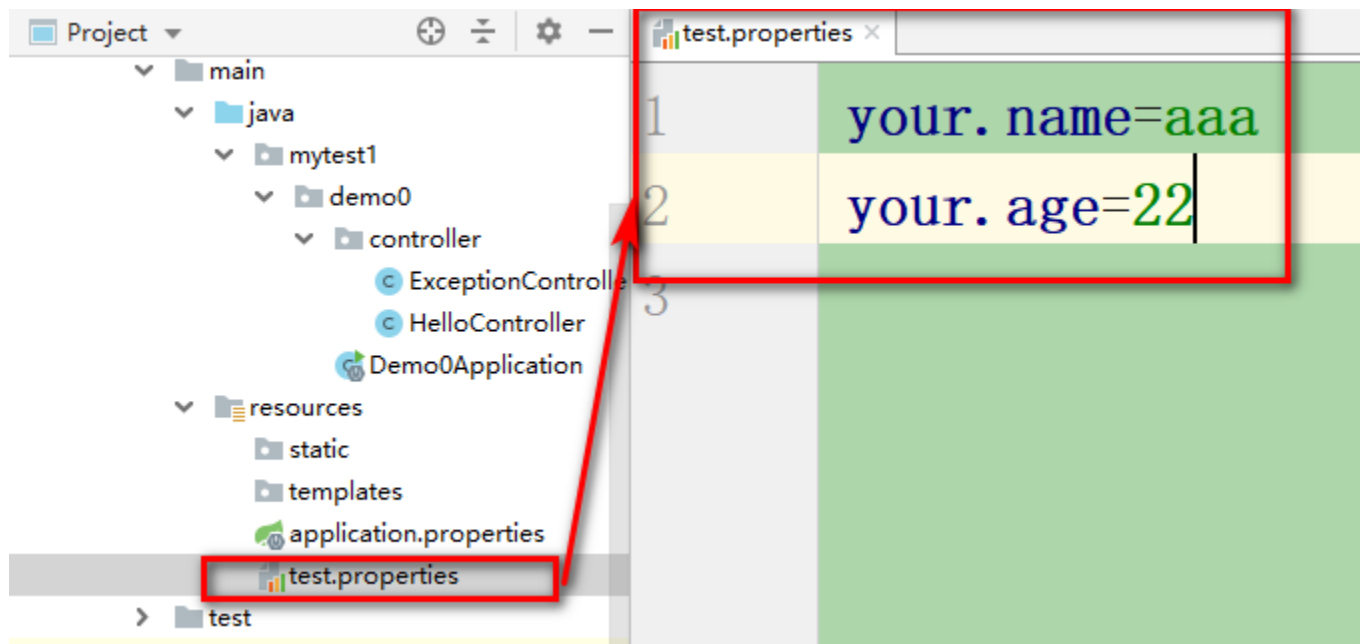
□实例

```
DemoApplication.java × application.yml × BootController.java ×  
1  spring:  
2    jpa:  
3      database: oracle  
4    datasource:  
5      url: jdbc:oracle:thin:@127.0.0.1:1521:orcl  
6      username: java05  
7      password: 1  
8      driver-class-name: oracle.jdbc.OracleDriver
```

自定义配置文件的配置及获取

□当我们自定义非spring配置文件时,可以使用
@PropertySource 注解加载自定义配置文件,然后就可
以使用@Value获取其中的属性了.

test.properties



Java代码

```
@RestController
```

```
@PropertySource("classpath:test.properties")
```

```
public class HelloController {
```

```
    @Value("${your.name}")
```

```
    private String name;
```

```
    @Value("${your.age}")
```

```
    private int age;
```

XML文件的配置及获取

□SpringBoot提倡无XML配置文件的方式进行框架的搭建,但如果仍想应用中仍然想采用以前 xml 文件的配置方式,如 "beans.xml", 则使用@ImportResource注解轻松搞定。

□建议非维护老项目,不要使用.

□ **@ImportResource** 标注在一个配置类上，通常直接放置在应用启动类上，和 **@SpringBootApplication** 一起即可。

```
4  /**
5   * 应用启动类
6   *
7   * @ImportResource 必须放置在配置类上，通常放在启动类即可，用 value 指明导入类路径下的那个 Spring 配置文件
8   */
9  @ImportResource(value = {"classpath:beans.xml"})
10 @SpringBootApplication
11 public class CocoApplication {
12     public static void main(String[] args) {
13         SpringApplication.run(CocoApplication.class, args);
14     }
15 }
```

□然后就可以在类路径下提供原始的 beans.xml 配置文件：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www
5
6       <!-- 放入到Spring容器中，这是以前Spring的内容，不再累述-->
7       <bean id="userService" class="com.lct.service.UserService"/>
8 </beans>
```