一、简介

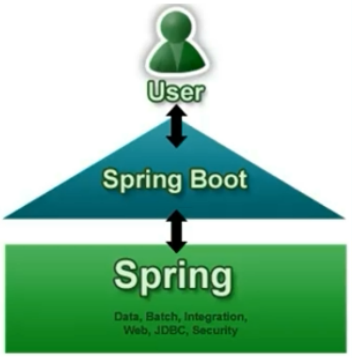
Spring Boot来简化Spring应用开发，约定大于配置，
去繁从简，just run就能创建一个独立的，产品级别的应用

背景：

J2EE笨重的开发、繁多的配置、低下的开发效率、复杂的部署流程、第三方技术集成难度大。

解决：

“Spring全家桶”时代。
Spring Boot → J2EE一站式解决方案
Spring Cloud → 分布式整体解决方案



- 优点：
  - 快速创建独立运行的Spring项目以及与主流框架集成
  - 使用嵌入式的Servlet容器，应用无需打成WAR包
  - starters自动依赖与版本控制
  - 大量的自动配置，简化开发，也可修改默认值
  - 无需配置XML，无代码生成，开箱即用
  - 准生产环境的运行时应用监控
  - 与云计算的天然集成

二、Spring Boot配置

主要介绍Spring Boot的配置文件、加载循序、配置原理

一、配置文件

- Spring Boot使用一个全局的配置文件
  - application.properties
  - application.yml

- 配置文件放在**src/main/resources**目录或者**类路径/config**下

- .yml是YAML（YAML Ain't Markup Language）语言的文件，以数据为中心，比json、xml等更适合做配置文件
  - http://www.yaml.org/ 参考语法规范

- 全局配置文件的可以对一些默认配置值进行修改

2、值的写法

字面量：普通的值（数字，字符串，布尔）

k: v：字面直接来写；

字符串默认不用加上单引号或者双引号；

""：双引号；不会转义字符串里面的特殊字符；特殊字符会作为本身想表示的意思

name: "zhangsan \n lisi"：输出；zhangsan 换行 lisi

''：单引号；会转义特殊字符，特殊字符最终只是一个普通的字符串数据

name: 'zhangsan \n lisi'：输出；zhangsan \n lisi

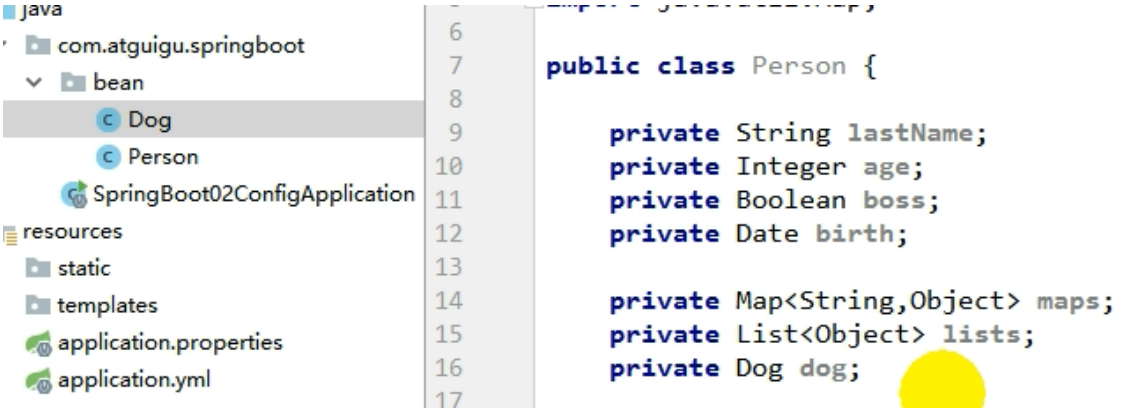
- 复合结构。以上写法的任意组合都是可以

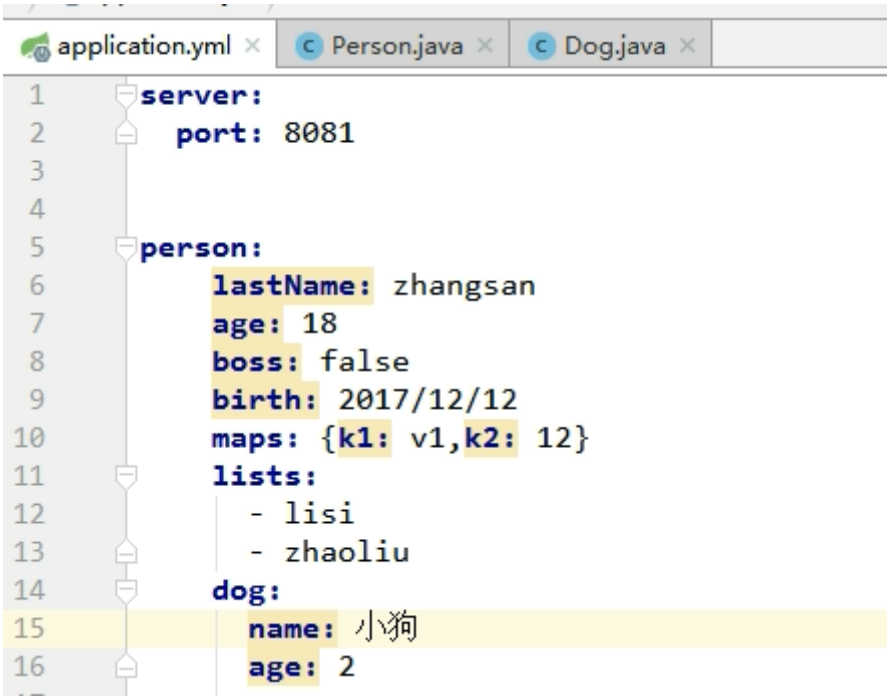
- 字面量
  - 数字、字符串、布尔、日期
  - 字符串
    - 默认不使用引号
    - 可以使用单引号或者双引号，单引号会转义特殊字符
    - 字符串可以写成多行，从第二行开始，必须有一个单空格缩进。换行符会被转为空格。
- 文档
  - 多个文档用 - - - 隔开

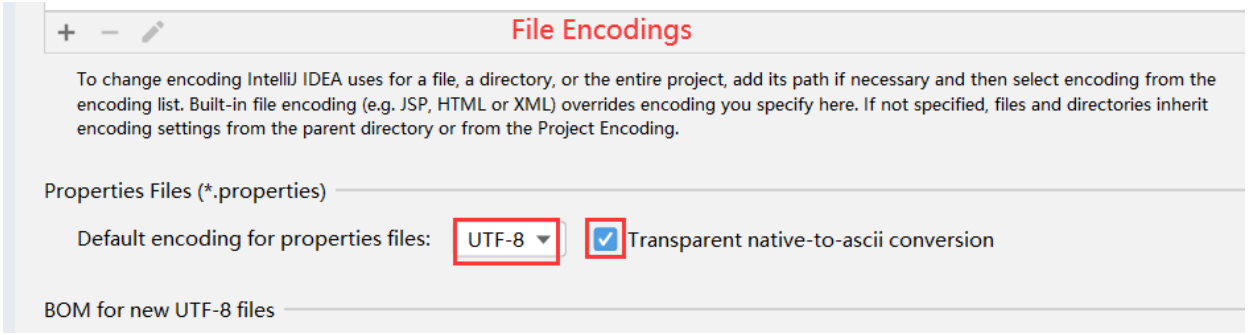
注意：

Spring Boot使用 snakeyaml 解析yml文件；

https://bitbucket.org/asomov/snakeyaml/wiki/Documentation#markdown-header-yaml-syntax 参考语法


java
com.atguigu.springboot
bean
Dog
Person
SpringBoot02ConfigApplication
resources
static
templates
application.properties
application.yml
public class Person {
private String lastName;
private Integer age;
private Boolean boss;
private Date birth;
private Map<String,Object> maps;
private List<Object> lists;
private Dog dog;


application.yml
Person.java
Dog.java
server:
port: 8081
person:
lastName: zhangsan
age: 18
boss: false
birth: 2017/12/12
maps: {k1: v1,k2: 12}
lists:
- lisi
- zhaoliu
dog:
name: 小狗
age: 2


File Encodings
To change encoding IntelliJ IDEA uses for a file, a directory, or the entire project, add its path if necessary and then select encoding from the encoding list. Built-in file encoding (e.g. JSP, HTML or XML) overrides encoding you specify here. If not specified, files and directories inherit encoding settings from the parent directory or from the Project Encoding.
Properties Files (*.properties)
Default encoding for properties files:
UTF-8
Transparent native-to-ascii conversion
BOM for new UTF-8 files

三、配置文件值注入

- **@Value和@ConfigurationProperties为属性注值对比**

Feature	`@ConfigurationProperties`	`@Value`
Relaxed binding	Yes	No
Meta-data support	Yes	No
`SpEL` evaluation	No	Yes

- **属性名匹配规则（Relaxed binding）**
  - person.firstName：使用标准方式
  - person.first-name：大写用-
  - person.first_name：大写用_
  - PERSON_FIRST_NAME：
    - 推荐系统属性使用这种写法

2、@Value获取值和@ConfigurationProperties获取值比较

	@ConfigurationProperties	@Value
功能	批量注入配置文件中的属性	一个个指定
松散绑定（松散语法）	支持	不支持
SpEL	不支持	支持
JSR303数据校验	支持	不支持
复杂类型封装	支持	不支持

配置文件yml还是properties他们都能获取到值；

如果说，我们只是在某个业务逻辑中需要获取一下配置文件中的某项值，使用@Value；

如果说，我们专门编写了一个javaBean来和配置文件进行映射，我们就直接使用@ConfigurationProperties；

- **@ConfigurationProperties**
  - 与@Bean结合为属性赋值
  - 与@PropertySource（只能用于properties文件）结合读取指定文件

- **@ConfigurationProperties Validation**
  - 支持JSR303进行配置文件值校验；

```
@ConfigurationProperties(prefix="connection")
@Validated
public class FooProperties {

    @NotNull
    private InetAddress remoteAddress;

    @Valid
    private final Security security = new Security();
```

- **@ImportResource读取外部配置文件**

@**ImportResource**：导入Spring的配置文件，让配置文件里面的内容生效；

Spring Boot里面没有Spring的配置文件，我们自己编写的配置文件，也不能自动识别；

想让Spring的配置文件生效，加载进来；@**ImportResource**标注在一个配置类上

```
@ImportResource(locations = {"classpath:beans.xml"})
导入Spring的配置文件让其生效
```

SpringBoot推荐给容器中添加组件的方式；推荐使用全注解的方式

1、配置类======Spring配置文件

2、使用@Bean给容器中添加组件

```
/**
 * @Configuration：指明当前类是一个配置类；就是来替代之前的Spring配置文件
 *
 * 在配置文件中用<bean><bean/>标签添加组件
 *
 */
@Configuration
public class MyAppConfig {

    //将方法的返回值添加到容器中；容器中这个组件默认的id就是方法名
    @Bean
    public HelloService helloService02(){
        System.out.println("配置类@Bean给容器中添加组件了...");
        return new HelloService();
    }
```

四、配置文件占位符

- **RandomValuePropertySource：配置文件中可以使用随机数**
  ${random.value}、${random.int}、${random.long}
  ${random.int(10)}、${random.int[1024,65536]}

- **属性配置占位符**

```
app.name=MyApp
app.description=${app.name} is a Spring Boot application
```

  - 可以在配置文件中引用前面配置过的属性（优先级前面配置过的这里都能用）。
  - ${app.name:默认值}来指定找不到属性时的默认值

```
# server.port=8081

# idea的properties配置文件使用utf-8编码，以前配置文件都是使用ascii编码
# 为了解决乱码需要在File Encoding中勾选设置Transparent native-to-ascii conversion
person.last-name=张三${random.uuid}
person.age=${random.int}
person.birth=2022/03/24
person.map.k1=v1
person.map.k2=v2
person.list=a,b,c
# ${person.hello:hello}没有person.hello属性，使用默认值hello
person.pet.name=${person.hello:hello}_miao
person.pet.age=1
```

五、 Profile

Profile是Spring对不同环境提供不同配置功能的支持，可以通过激活、指定参数等方式快速切换环境

1、多profile文件形式：
   - 格式：application-{profile}.properties：
     - application-dev.properties、application-prod.properties

2、多profile文档块模式：

3、激活方式：
   - 命令行 --spring.profiles.active=dev
   - 配置文件 spring.profiles.active=dev
   - jvm参数 –Dspring.profiles.active=dev

```yaml
spring:
  profiles:
    active: prod # profiles.active: 激活指定配置
---
spring:
  profiles: prod
server:
  port: 80
--- #三个短横线分割多个profile区（文档块）
spring:
  profiles: default # profiles: default表示未指定默认配置
server:
  port: 8080
```

六、配置文件加载位置

spring boot 启动会扫描以下位置的application.properties或者application.yml文件作为Spring boot的默认配置文件

- file:./config/
- file:./
- classpath:/config/
- classpath:/
- 以上是按照**优先级从高到低**的顺序，所有位置的文件都会被加载，**高优先级配置**内容会**覆盖低优先级配置**内容。
- 我们也可以通过配置spring.config.location来改变默认配置

maven打包命令不会打包项目路径下的配置文件：-file:./config/和-file:./都不会打包

七、外部配置加载顺序

Spring Boot 支持多种外部配置方式

这些方式优先级如下：
https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#boot-features-external-config

1. 命令行参数
2. 来自java:comp/env的JNDI属性
3. Java系统属性（System.getProperties()）
4. 操作系统环境变量
5. RandomValuePropertySource配置的random.*属性值
6. jar包外部的application-{profile}.properties或application.yml(带spring.profile)配置文件
7. jar包内部的application-{profile}.properties或application.yml(带spring.profile)配置文件
8. jar包外部的application.properties或application.yml(不带spring.profile)配置文件
9. jar包内部的application.properties或application.yml(不带spring.profile)配置文件
10. @Configuration注解类上的@PropertySource
11. 通过SpringApplication.setDefaultProperties指定的默认属性

由jar包外向jar包内进行寻找；

优先加载带profile

6.jar包外部的application-{profile}.properties或application.yml(带spring.profile)配置文件

7.jar包内部的application-{profile}.properties或application.yml(带spring.profile)配置文件

再来加载不带profile

8.jar包外部的application.properties或application.yml(不带spring.profile)配置文件

9.jar包内部的application.properties或application.yml(不带spring.profile)配置文件