

专题二

Spring Boot应用

——热部署及配置文件

- **热部署**

- 热部署概念
- 热部署原理
- 热部署配置步骤

- **配置文件**

- 配置文件概述
- 配置文件创建
- 常用配置项
- 配置项读取

热部署简介

- 所谓热部署，就是在应用正在运行的时候升级软件，却不需要重新启动应用。
- 对于Java应用程序来说，热部署就是在运行时更新Java类文件，不需要重启服务即可生效。
- 热部署可以大大提高我们的开发效率。

热部署原理

- 热部署中有二个类启动器
 - baseclassloader : 加载不会变化的类, 比如第三方的jar
 - restartclassloader : 加载变化的类, 基本是classpath下的类
- 在热部署时, base类加载器不会重新加载, 而restart会被废弃, 由新的restart来代替, 所以速度看起来被重启要快一些。

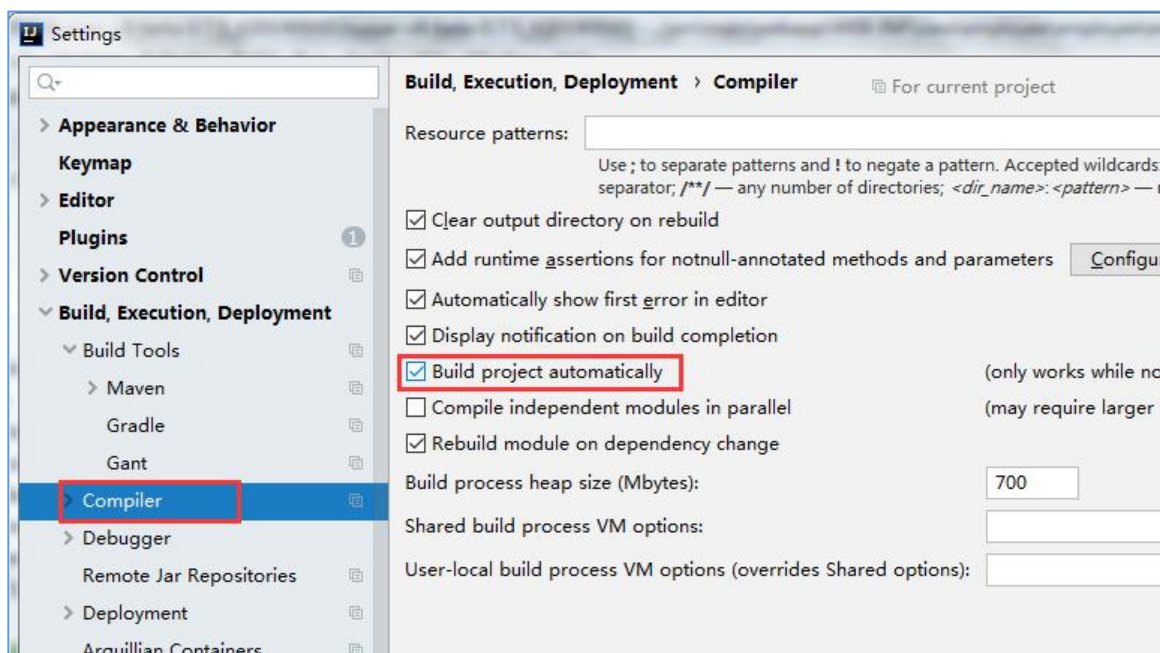
01_添加依赖

- 在pom.xml中添加热部署依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional>
</dependency>
```

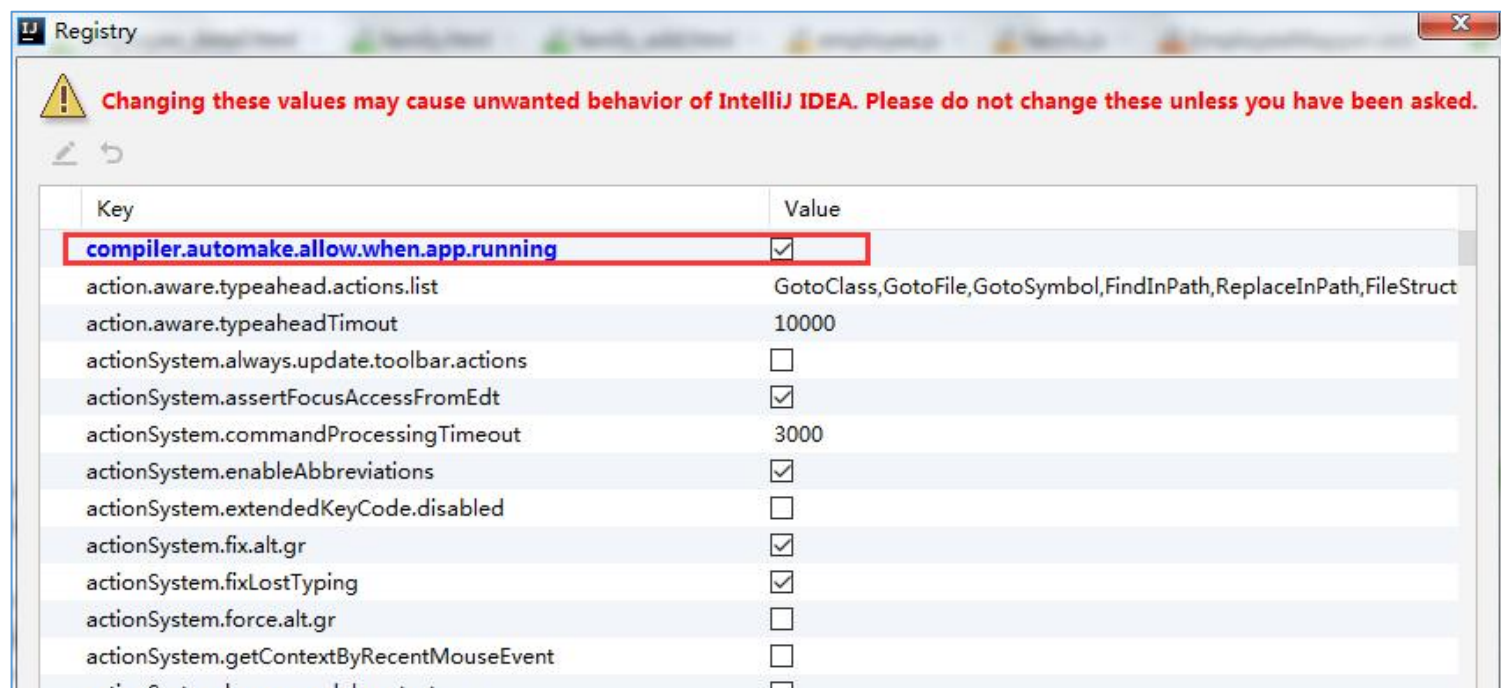
02_设置idea自动编译

- 选择idea菜单的file——>Settings——>builder——>compiler ,
- 右侧勾选Build project automatically



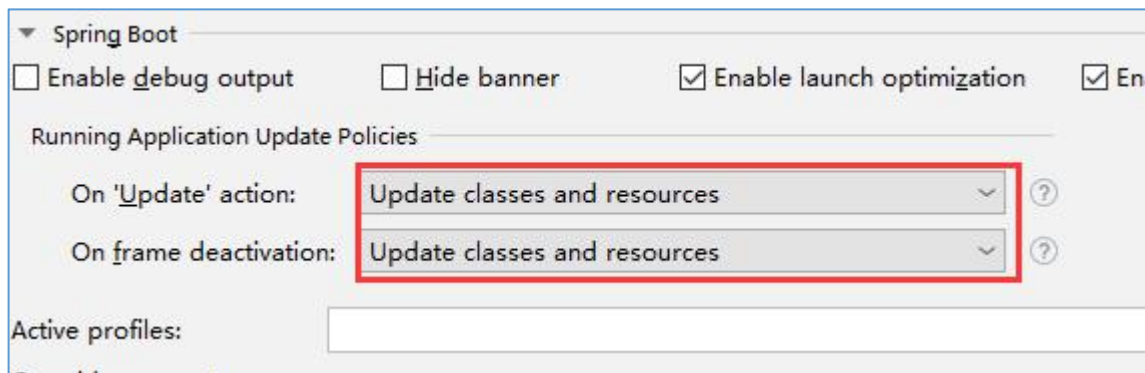
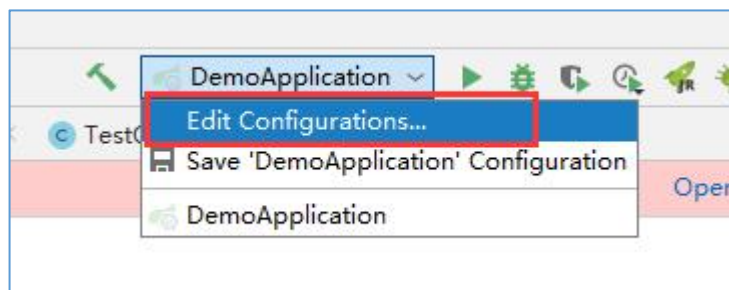
03_设置idea自动编译

- 同时按住shift+ctrl+alt+/,打开注册功能
- 勾选compiler.automake.allow.when.app.running



04_服务启动配置

- 注意：以上操作完成不生效的话，才需要该步骤



05_热部署测试

- 以上操作完成之后，热部署即可生效，当我们进行代码修改时，不再需要手动重启。

```
2019-12-09 16:17:14.505 INFO 5704 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWel
2019-12-09 16:17:14.521 INFO 5704 --- [ restartedMain] com.jxd.demo.DemoApplication
2019-12-09 16:17:59.823 INFO 5704 --- [ Thread-19] o.s.s.concurrent.ThreadPoolTaskEx
```

```

      _ _ _ _ _
     /\ / _ _ _ _ _
    ( () \_ | ' _ | ' _ | ' _ \_ | \_ \_ \_ \
   \V _ _ | | | | | | | | | | | | | | | | | |
    ' _ _ | . _ | | | | | | | | | | | | | | |
   =====|_|=====|_|/=//_/_/_/_/
:: Spring Boot ::      (v2.1.5.RELEASE)
```

```
2019-12-09 16:18:00.153 INFO 5704 --- [ restartedMain] com.jxd.demo.DemoApplication
```

为什么需要配置文件


- **springboot是基于约定开发的，约定大于配置**
 - SpringBoot提供了大量的通用配置项来满足我们的开发
 - 开发中的常规项，都在底层帮我们自动配置好，都有默认值，比如默认端口等
- **开发中的个性化配置**
 - 但是真正的开发中我们会有个性化的需求，默认配置项不再能满足
 - 我们需要在配置文件中进行个性化的需求配置
- **我们的项目会有一些信息要告诉springboot**
 - 我们的数据库所在地址，数据库的用户名密码
 - 我们可能会改变模板路径
 - 比如我们写的映射文件的路径也需要告诉springboot

配置文件类型

- SpringBoot规定：我们可以在resources目录下，自定义文件名为application*.yaml、application*.yml以及application*.properties的配置文件,其中*符号匹配任意字符。
- 三类文件从上往下依次加载，所以第三种配置文件优先级最高。
- 我们主要学习应用yaml类型，在resources创建application.yml文件

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>

  <resource>
    <directory>${basedir}/src/main/resources</directory>
    <excludes>
      <exclude>**/application*.yml</exclude>
      <exclude>**/application*.yaml</exclude>
      <exclude>**/application*.properties</exclude>
    </excludes>
  </resource>
```



查看发现

YAML简介

- 在yml以及yaml类型的配置文件中，主要应用YAML语言。
- YAML是一种标记语言，这个语言是通用的，C++、python、php、C、Java等都识别。
- 它通过树形结构来展示数据，可读性、易读性更高
 - 同一个配置项下面的同缩进的项都是一个层级的，如下，servlet和port属于同级别

```
##### 项目启动信息配置
server:
  servlet:
    context-path: /
  port: 8080
```

YAML语法

- 01_YAML通过key-value设置属性值
 - key : value
 - value值之前必须有一个空格
- 02_# 代表注释内容
- 03_YAML属于大小写敏感的语言，所以要严格区分大小写

```
#定义简单数据的配置项  
name: 金桥工程
```

常用数据配置项

- 在application.yml中定义简单数据项

#定义简单数据的配置项

name: 金桥工程

#定义对象

person:

name: 李丽

addr: 济南

#定义数组和list数据

student:

- name: 王朋 //数组定义需要-符号

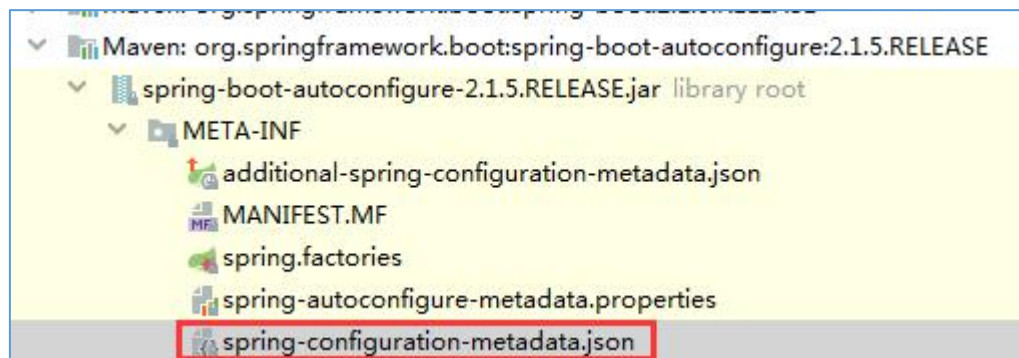
addr: 北京

- name: 赵辉

addr: 济南

SpringBoot默认配置项

- SpringBoot提供了很多默认配置项为我们的开发提供支持
- SpringBoot的默认配置项存放位置：
- 在依赖包在spring-boot-autoconfigure包下
 - 》 META-INF
 - 》 spring-configuration-metadata.json



SpringBoot默认配置项

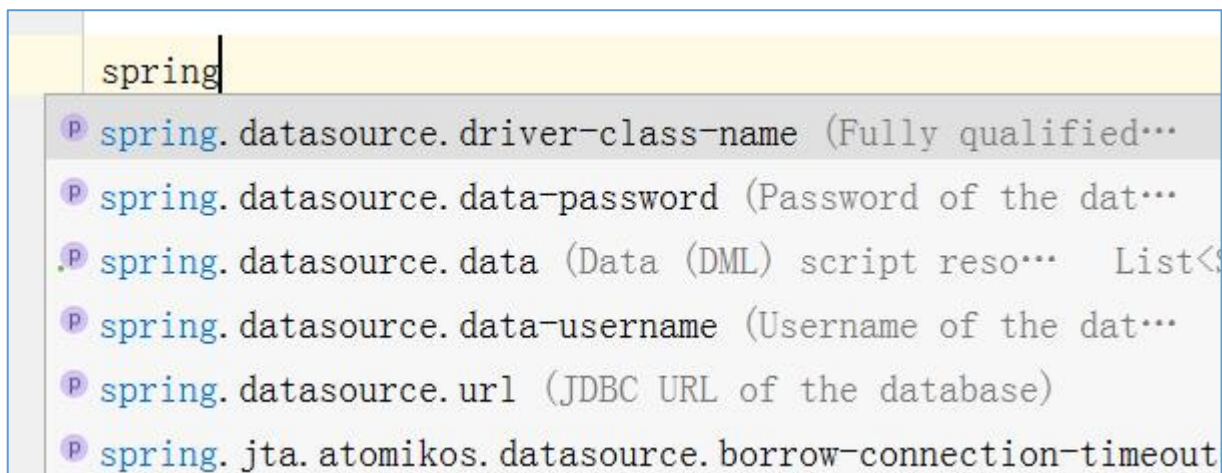
在json文件中，name值对应配置项的名称，defaultValue为SpringBoot提供的默认值，如：服务端口默认为8080

- 1) 可以通过ctrl+F搜索需要设置的配置项
- 2) 可以直接复制name的值粘贴到配置文件中进行自定义设置，该属性会自动转换为树形结构



自定义配置项

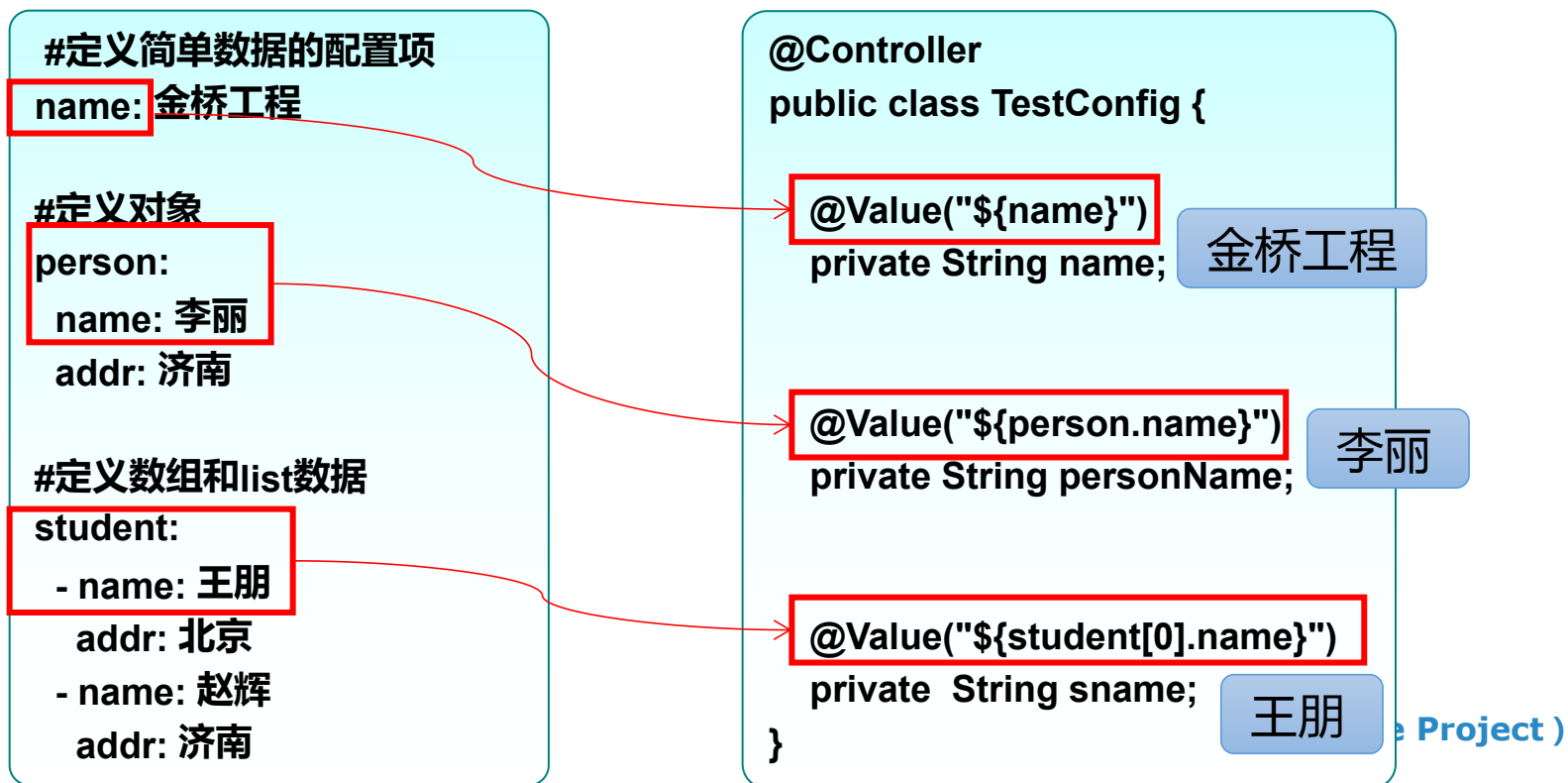
- 我们也可以在application.yml文件中直接编辑，进行相关配置，如果我们输入的内容是默认配置中包含的，那么会有对应的补全提示，选中要设置的内容回车即可。



- 如果是自定义的其他配置项，则不会有对应的提示，比如我们上面定义的数据配置项或者是beetl相关内容。

配置项读取

- 如果需要在业务代码中获取配置项的值，可以使用以下两种方式：
- 方式一：`@Value("${yml文件中的对象属性的名字}")`，注解在要注入值的属性上
 - 这种方式适用于配置项较少时



配置项读取

- 方式二：使用@ConfigurationProperties(prefix = "person")注解一个类
 - 》这个注解会把prefix前缀对应的配置项下的所有属性获取到
 - 》类中属性名称和配置项中名称要一一对应
 - 》并且提供get/set方法
 - 该方式适用于配置项较多时

@Controller

//prefix代表就是配置项的顶级名称

@ConfigurationProperties(prefix = "person")

public class ConfigYml {

//属性名称要和配置项名称一致

private String name;
private String addr;

public String getName() { return name; }
public void setName(String name) { this.name = name; }
public String getAddr() { return addr; }
public void setAddr(String addr) { this.addr = addr; }

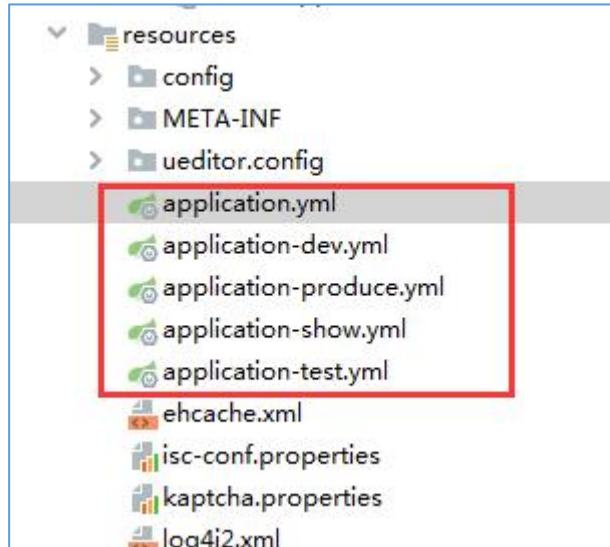
#定义对象

person:

name: 李丽
addr: 济南

项目中的配置文件

- 配置文件的主要目的就是针对我们的项目进行个性化的配置，我们可以在resources目录下创建多个配置文件，去完成项目配置。
- 项目启动时，会读取这些配置，从而完成项目的构建。



```
##### spring配置 开始 #####
spring:
  main:
    allow-bean-definition-overriding: true
  profiles:
    active: dev
  mvc:
    static-path-pattern: /static/** # 静态资源路径
  view:
    prefix: /WEB-INF/view # 页面文件路径,
  http:
    converters:
      preferred-json-mapper: fastjson # Json转换器
  servlet:
    multipart:
      max-request-size: 100MB # 最大请求大小
      max-file-size: 100MB # 最大文件大小
  devtools:
```