SpringBoot基础配置

properties 文件

语法

> key=value

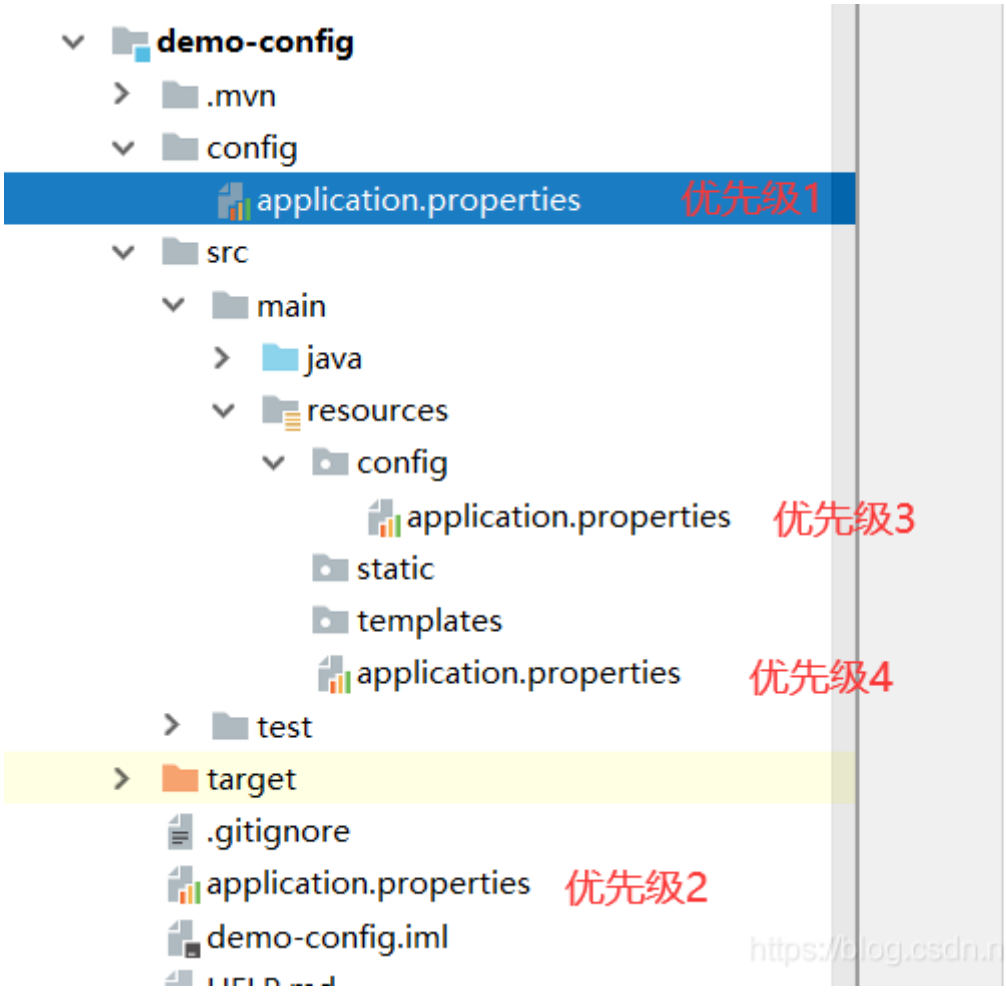
比如配置Tomcat的端口：server.port=8081。对大小写敏感。

优先级

一般情况下，properties可以在项目的4个地方存在，然后他们每个位置有不同的优先级，且不同名的时候application.properties的优先级是最高的。

扫描的包路径按照优先级有下面四个：

> - file:./config/
> - file:./
> - classpath:/config/
> - classpath:/



自定义数据配置

自定义对象数据：

```
personal.name = "测试"
personal.age = 21
personal.friends = "开发1", "开发2", "开发3"
```

如果遇到java的驼峰命名，在properties文件里面的命名就比较宽松，比如java里面用studentName，我在properties里面既可以用studentName，也可以用student-name，或者是student_name,都行。

自定义的实体类：

```
@Component
@ConfigurationProperties(prefix = "personal")
public class Personal {

    @Value("${personal.name}")
    private String name;

    private Integer age;

    private List<String> friends;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public List<String> getFriends() {
        return friends;
    }
```

```
    public void setFriends(List<String> friends) {
        this.friends = friends;
    }
}
```

controller测试代码：

```
@Controller
@ResponseBody
public class PersonalController {

    @Autowired
    Personal personal;

    @RequestMapping("/personal")
    public Personal getPersonal() {
        return personal;
    }
}
```

yml文件

语法

> key空格:空格value

注意：空格不能少，大小写敏感
利用缩进代表层级关系，只要是缩进一样就代表是一级的

优先级

在既有yml文件也有properties文件的时候（其实还有一种yaml，和yml差不多的），yml加载顺序是先于properties的，所以优先级是properties大于yml，然后其他的情况下的优先级和properties文件是一样的。

自定义数据配置

```
personal:
  name: 开发
  age: 20
  friends: [测试1,  测试2]
```

获取自定义数据配置

> 同properties文件

多配置文件使用

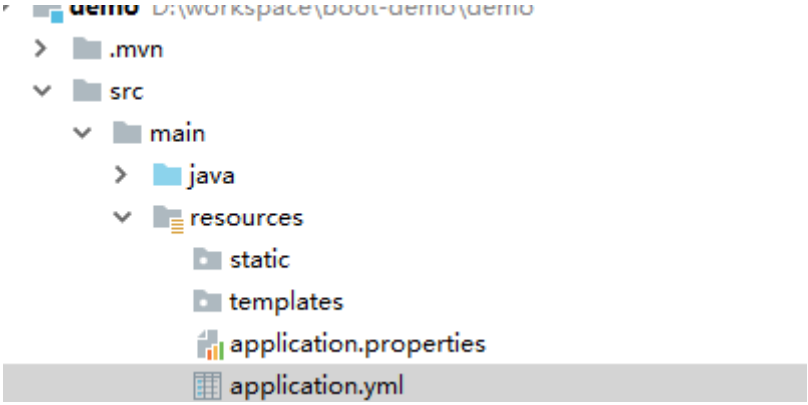
```
application.yml
spring:
  profiles:
    active: dev

application-dev.yml
server:
  port: 8088
```

web静态资源配置

项目创建

创建一个空的项目，项目的依赖配置为**starter-web**依赖，创建好的项目下面有一个**resources**文件夹，里面有一些空的默认的文件夹，然后有一个配置文件。



资源文件访问

- Springboot中默认的静态资源路径：

> 1. classpath:/static,
> 2. classpath:/public,
> 3. classpath:/resources,
> 4. classpath:/META-INF/resources,

classpath 在项目中就相当于src/main/resources文件夹.

- 自定义静态资源路径

```
spring.resources.static-locations=classpath:templates/
```

配置了静态资源路径映射之后，**我们只能访问这个路径下面的资源**，也就是相当于自定义了静态资源路径，**就是说默认的静态资源路径都会失效。**

其他静态资源路径配置方式

继承WebMvcConfigurerAdapter

```
@Configuration
public class WebConfig extends WebMvcConfigurerAdapter {
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {

 registry.addResourceHandler("/**").addResourceLocations("/","classpath:t
emplates/");
        super.addResourceHandlers(registry);
    }
}
```

优化配置

```
#加前缀
spring.mvc.view.prefix=/
#加后缀
spring.mvc.view.suffix=.html
```

通过以上的配置，我们之前的controller里面的方法返回就可以写成这样。

```
@RequestMapping("/getHtml")
public String getHtml(){
    return "index";
}
```

总结:

静态资源配置路径的两种常用方式： 一种是默认配置，一种是自定义配置，自定义配置可以是代码配置，可以是配置文件里面配置。这两种配置方式都会使原来默认的配置方式**失效**。

静态资源的访问方式有两种：一种是自己用代码定义路径，然后访问，一种是直接利用特定的URL方式访问静态资源。