Spring Boot

使用springboot就不需要mybatis.xml,spring.xml,springmvc.xml三个配置文件了

通过 Spring Boot 可以快速构建一个基于 Spring 框架的 Java Application，简化配置，**自动装配**。

JavaConfiguration 用 Java 类替代 XML 的配置方式。

Spring Boot 对常用的第三方库提供了配置方案，可以很好地和 Spring 进行整合，一键式搭建功能完备的 Java 企业级应用。

开箱即用是 Spring Boot 的特点

Spring Boot 的优势：

- 不需要任何 XML 配置文件
- 内嵌 Tomcat，可以直接部署
- 默认支持 JSON 数据，不需要进行转换
- 支持 RESTful
- 配置文件非常简单，支持 YAML 格式

Spring Boot 是一种只需要极少配置就可以快速搭建 Spring 应用，并且集成了常用的第三方类库，让开发者可以快速进行企业级应用开发。

Spring Boot 2.x 要求必须基于 Spring 5.x，Spring 5.x 要求 Java 版本必须是 8 以上。

Spring Boot 的使用

1、创建 Handler

```
package com.southwind.springboot.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/hello")
public class HelloHandler {

    @GetMapping("/index")
    public String index(){
        return "Hello Spring Boot";
    }
}
```

@RestController是@Controller和@ResponseBody的结合体,两个标注合并起来的作用。

GetMapping

2、创建启动类

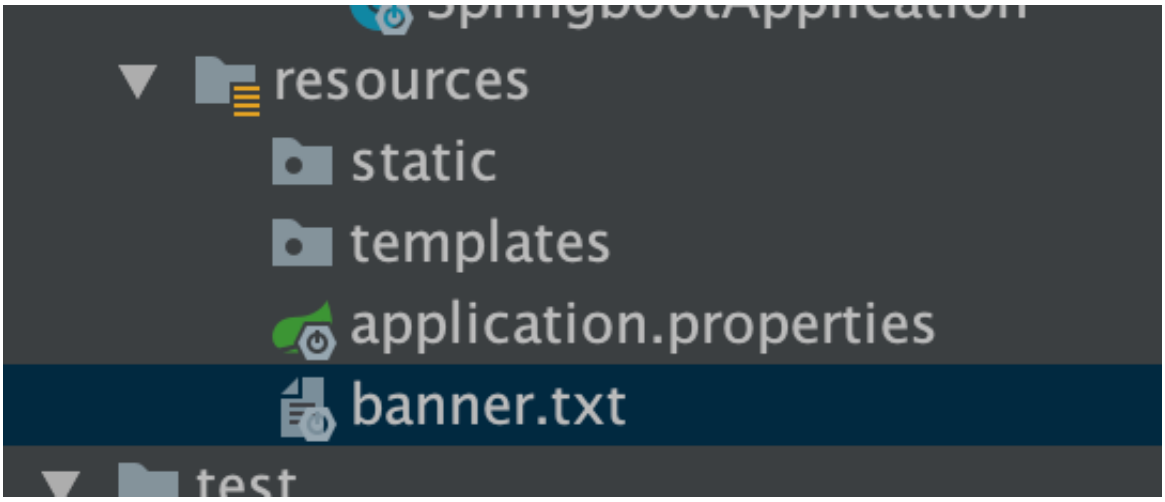
```
package com.southwind.springboot.controller;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/hello")
public class HelloHandler {

    @GetMapping("/index")
    public String index(){
        return "Hello Spring Boot";
    }
}
```

启动类必须覆盖所有与业务相关的类：启动类所在的包必须是业务类所在包的同包或者父包，如果没有覆盖，业务类就不会自动装配到 IoC 容器中。

Spring Boot 配置文件 springboot是不需要xml配置文件但还是有配置文件的

自定义 banner



Properties

```
#端口
server.port=8181
#项目访问路径
server.servlet.context-path=/springboot
#cookie失效时间
server.servlet.session.cookie.max-age=100
#session失效时间
server.servlet.session.timeout=100
#编码格式
server.tomcat.uri-encoding=UTF-8
```

YAML

YAML 是不同于 Properties 的另外一种文件格式，同样可以用来写配置文件，Spring Boot 默认支持 YAML 格式，YAML 的优点在于编写简单，结构清晰，利用缩紧的形式来表示层级关系。

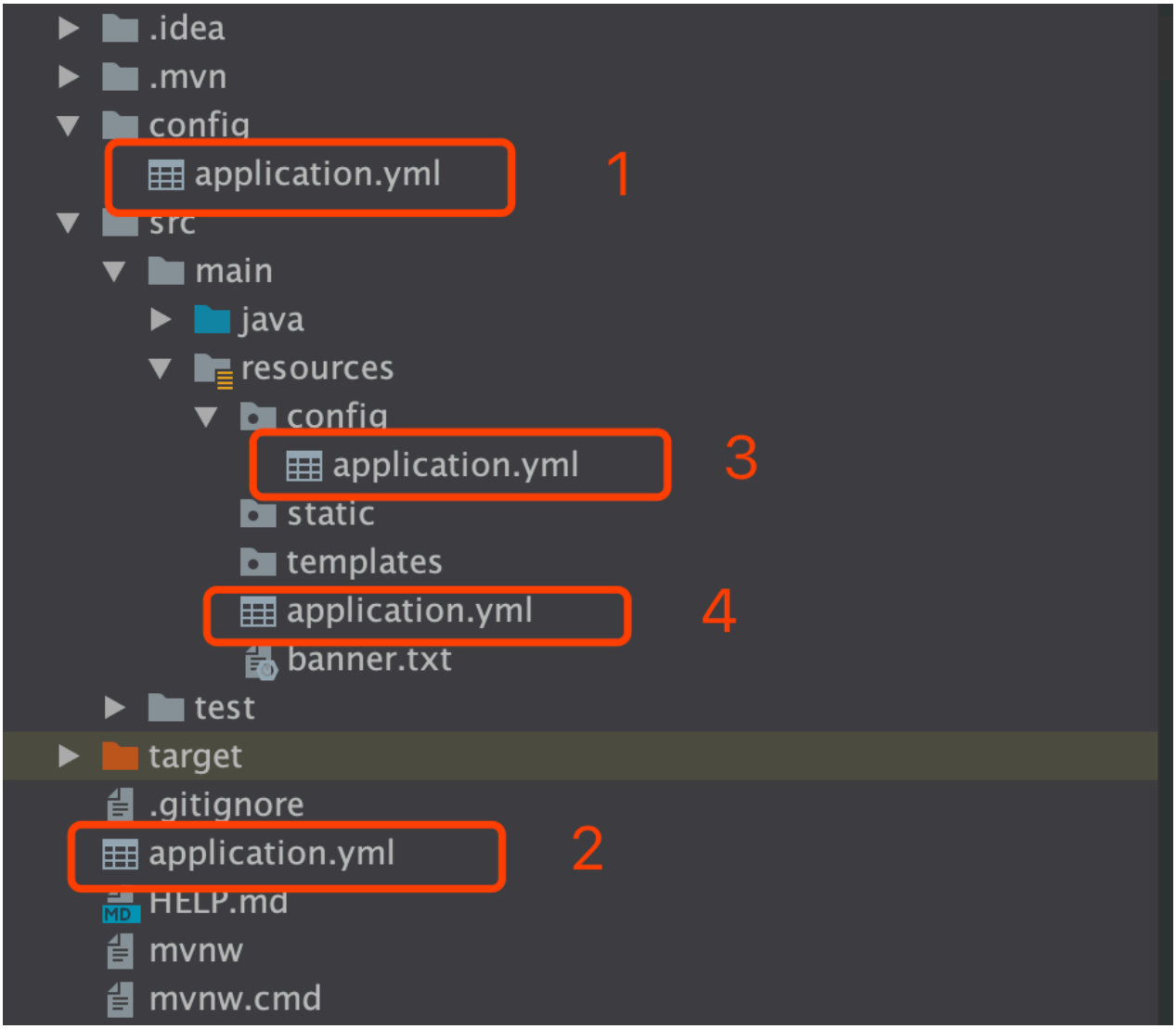
相比于 Properties，YAML 可以进一步简化配置文件的编写，更加方便。

```yaml
server:
  port: 8181
  servlet:
    context-path: /springboot
    session:
      cookie:
        max-age: 100
      timeout: 100
  tomcat:
    uri-encoding: UTF-8
```

需要注意的是 YAML 格式书写规范非常严格，属性名和属性值之间必须至少一个空格。

如果 Properties 和 YAML 两种类型的文件同时存在，Properties 的优先级更高。

配置文件除了可以放置在 resources 路径下之外，还有 3 个地方可以放置，如下图所示。



优先级顺序如下所示：

1、根路径下的 config 中的配置文件

2、根路径下的配置文件

3、resources 路径下的 config 中的配置文件

4、resources 路径下的配置文件

可以直接在 Handler 中读取 YAML 文件中的数据，比如在业务方法中向客户端返回当前服务的端口信息。

```
package com.southwind.springboot.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/hello")
public class HelloHandler {

    /**
     * SpEL Spring Expression Language
     */
    @Value("${server.port}")
    private String port;

    @GetMapping("/index")
    public String index(){
        return "当前服务的端口是："+this.port;
    }
}
```

@Value 注解同样适用于 Properties 文件。

Spring Boot 整合 JSP

Spring Boot 与视图层的整合

- JSP
- Thymeleaf

Java Server Page，是 Java 提供的一种动态网页技术，底层是 Servlet，可以直接在 HTML 中插入 Java 代码。

JSP 底层原理：

JSP 是一种中间层组件，开发者可以在这个组件中将 Java 代码与 HTML 代码进行整合，由 JSP 引擎将组件转为 Servlet，再把开发者定义在组件中的混合代码翻译成 Servlet 的响应语句，输出给客户端。

1、创建基于 Maven 的 Web 项目，pom.xml

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.2.4.RELEASE</version>
</parent>

<dependencies>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>2.2.4.RELEASE</version>
  </dependency>

  <dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <version>9.0.19</version>
  </dependency>

</dependencies>
```

2、创建 Handler

```
package com.southwind.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
@RequestMapping("/hello")
public class HelloHandler {

    @GetMapping("/index")
    public ModelAndView index(){
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("index");
        modelAndView.addObject("mess","Hello Spring Boot");
        return modelAndView;
    }
}
```

3、JSP

```
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>Index</h1>
    ${mess}
</body>
</html>
```

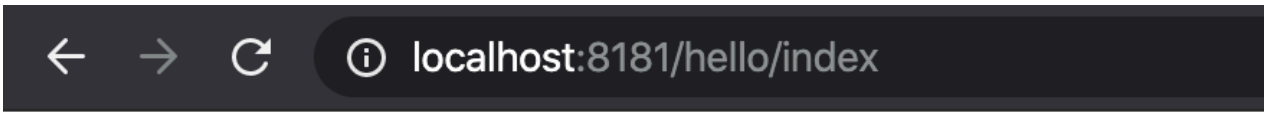
4、application.yml

```
server:
  port: 8181
spring:
  mvc:
    view:
      prefix: /
      suffix: .jsp
```

5、Application

```
package com.southwind;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class,args);
    }
}
```

看到如下界面，说明访问成功



Index

Hello Spring Boot

自动装备的关键

自动配置包，可以根据包自动把组件装到IOC

@SpringBootApplication

...

@SpringBootConfigration

@EnableAutoConfigration

@ComponentScan

...

@AutoConfigurationPackage

...

@Import({Registrar.class})

Registrar会返回一些bean的信息，import会读取然后装到IOC容器

registrar类中会把

配置了@SpringBootApplication的默认扫描包下的给扫描，所以启动类所在的包必须是业务类所在包的同包或者包