Spring Boot

仍然需要进行大量的配置，开发效率不高

专注于业务逻辑的开发，而非业务之外的配置，自动完成

使用 Spring Boot 来构建 Spring 项目，帮助开发者省略掉各种配置，提高开发效率

Spring Boot 底层实现，自动装配机制，Starter 启动器

Spring Boot 去掉各种繁琐的配置文件，但是为什么 Spring Boot 仍然需要 application.yml?

去掉都是通用的配置，个性化的配置必须开发者自己来配的

创建 Spring Boot 的三种方式

1、Maven 工程，手动配置依赖，创建配置文件，创建启动类

2、在线创建 Spring Boot 工程，Spring 提供在线代码生成工具，可以直接下载代码

3、IDEA 在线创建

整合 MyBatis

1、pom.xml 引入相关依赖

```
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>2.2.2</version>
</dependency>

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
```

2、application.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/dbname
    username: root
    password: root
  mybatis:
    mapper-locations: classpath:/mapping/*.xml
```

3、创建实体类

```java
package com.southwind.entity;

import lombok.Data;

@Data
public class Account {
    private Integer id;
    private String name;
}
```

4、Mapper 接口

```java
package com.southwind.entity;

import lombok.Data;

@Data
public class Account {
    private Integer id;
    private String name;
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.southwind.mapper.AccountMapper">

    <select id="list" resultType="com.southwind.entity.Account">
        select * from account
    </select>

</mapper>
```

5、创建控制器

```java
package com.southwind.controller;

import com.southwind.entity.Account;
import com.southwind.mapper.AccountMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class AccountController {

    @Autowired
    private AccountMapper accountMapper;

    @GetMapping("/list")
    public List<Account> list(){
        return this.accountMapper.list();
    }
```

```
}
```

6、启动类添加扫包配置

```
package com.southwind;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@MapperScan("com.southwind.mapper")
public class SpringbootO02DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringbootO02DemoApplication.class, args);
    }

}
```

整合 Thymeleaf

替代 JSP，效率更高的一种视图层解决方案

1、pom.xml 添加依赖

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

2、application.yml 配置视图解析器

```
spring:
  thymeleaf:
    prefix: classpath:/templates/
    suffix: .html
```

3、控制器

```
package com.southwind.controller;

import com.southwind.mapper.AccountMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class AccountController {

    @Autowired
    private AccountMapper accountMapper;
```

```
    @GetMapping("/list")
    public ModelAndView list(){
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("index");
        modelAndView.addObject("list", this.accountMapper.list());
        return modelAndView;
    }

}
```

4、创建 html

```
<!DOCTYPE html>
<html lang="en">
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1>用户信息</h1>
    <table>
        <tr>
            <td>编号</td>
            <td>姓名</td>
        </tr>
        <tr th:each="account:${list}">
            <td th:text="${account.id}"></td>
            <td th:text="${account.name}"></td>
        </tr>
    </table>
</body>
</html>
```

> 常用标签

th:text 用于文本显示

th:if 用于条件判断，条件成立则显示内容，否则不显示内容

th:each 用于遍历集合

th:value 用来给标签赋值，给拥有 value 属性的标签进行赋值