Spring Boot

在 Spring MVC 体系中虽然简化了配置，减少了类的数量，一个控制器中多个方法可以分别对应不同的业务。但是从根本上看，需要的配置还是太多，搭建工程还是会有很多重复性的步骤，开发速度还是不够快，应该进一步简化，Spring Boot 就应运而生，一个 Spring 提供的快速开发框架。

Spring Boot 开启了各种自动装配，就是为了简化开发，不需要写各种配置文件，只需要引入相关依赖就能快速搭建一个 Web 工程。

特点

- 不需要 web.xml 配置。
- 不需要 springmvc.xml 配置。
- 不需要 tomcat，Spring Boot 内嵌了 tomcat。
- 不要配置 JSON 解析，直接支持 RESTful 风格。
- 个性化配置时，最少一个配置文件就可以配置所有的个性化信息。

如何使用

- 创建 Maven 工程，pom.xml

```
<!-- 继承父包 -->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.7.RELEASE</version>
</parent>

<dependencies>
    <!-- Spring Boot web 启动包 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.6</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

- 创建实体类 Student

```java
package com.southwind.entity;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class Student {
    private long id;
    private String name;
    private int age;
}
```

- 创建 StudentRepository 接口以及实现类 StudentRepositoryImpl

```java
package com.southwind.repository;

import com.southwind.entity.Student;

import java.util.Collection;

public interface StudentRepository {
    public Collection<Student> findAll();
    public Student findById(long id);
    public void save(Student student);
    public void update(Student student);
    public void deleteById(long id);
}
```

```java
package com.southwind.repository.impl;

import com.southwind.entity.Student;
import org.springframework.stereotype.Repository;

import java.util.Collection;
import java.util.HashMap;
import java.util.Map;

@Repository
public class StudentRepositoryImpl implements
com.southwind.repository.StudentRepository {
    private static Map<Long,Student> studentMap;

    static {
        studentMap = new HashMap<>();
        studentMap.put(1L,new Student(1L,"张三",22));
```

```
        studentMap.put(2L,new Student(2L,"李四",23));
        studentMap.put(3L,new Student(3L,"王五",24));
    }

    @Override
    public Collection<Student> findAll() {
        return studentMap.values();
    }

    @Override
    public Student findById(long id) {
        return studentMap.get(id);
    }

    @Override
    public void save(Student student) {
        studentMap.put(student.getId(),student);
    }

    @Override
    public void update(Student student) {
        studentMap.put(student.getId(),student);
    }

    @Override
    public void deleteById(long id) {
        studentMap.remove(id);
    }
}
```

- 创建 StudentService 接口以及实现类 StudentServiceImpl

```
package com.southwind.service;

import com.southwind.entity.Student;

import java.util.Collection;

public interface StudentService {
    public Collection<Student> findAll();
    public Student findById(long id);
    public void save(Student student);
    public void update(Student student);
    public void deleteById(long id);
}
```

```java
package com.southwind.service.impl;

import com.southwind.entity.Student;
import com.southwind.repository.StudentRepository;
import com.southwind.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Collection;

@Service
public class StudentServiceImpl implements StudentService {

    @Autowired
    private StudentRepository studentRepository;

    @Override
    public Collection<Student> findAll() {
        return studentRepository.findAll();
    }

    @Override
    public Student findById(long id) {
        return studentRepository.findById(id);
    }

    @Override
    public void save(Student student) {
        studentRepository.save(student);
    }

    @Override
    public void update(Student student) {
        studentRepository.update(student);
    }

    @Override
    public void deleteById(long id) {
        studentRepository.deleteById(id);
    }
}
```

- 创建 StudentHandler

```java
package com.southwind.controller;

import com.southwind.entity.Student;
import com.southwind.service.StudentService;
```

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.Collection;

@RestController
@RequestMapping("/student")
public class StudentHandler {

    @Autowired
    private StudentService studentService;

    @GetMapping("/findAll")
    public Collection<Student> findAll(){
        return studentService.findAll();
    }

    @GetMapping("/findById/{id}")
    public Student findById(@PathVariable("id") long id){
        return studentService.findById(id);
    }

    @PostMapping("/save")
    public void save(@RequestBody Student student){
        studentService.save(student);
    }

    @PutMapping("/update")
    public void update(@RequestBody Student student){
        studentService.update(student);
    }

    @DeleteMapping("/deleteById/{id}")
    public void deleteById(@PathVariable("id") long id){
        studentService.deleteById(id);
    }
}
```

- 在 resources 路径下创建配置文件 application.yml

```yaml
server:
  port: 9090
```

- 创建启动类 Main，Spring Boot 的入口类。

```
package com.southwind;

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

@SpringBootApplication
public class Main {

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

所有的业务相关的类必须放置在 Main 类所在包中，或者其子包中，才能完成自动装配。

RESTful 具体来讲就是 HTTP 协议中的四个表示动作的动词：GET、POST、PUT、DELETE，分别对应四种基本操作 CRUD。

GET 用来表示获取资源。

POST 用来表示新建资源。

PUT 用来表示修改资源。

DELETE 用来表示删除资源。

HTTP URL 含义

GET `http://localhost:9090/student/findAll` 获取所有学生对象

GET `http://localhost:9090/student/findById/1` 获取 id = 1 的学生对象

PSOT `http://localhost:9090/student/save` 添加学生对象

PUT `http://localhost:9090/student/update` 修改学生对象

DELETE `http://localhost:9090/student/deleteById` 删除 id = 1 的学生对象

Spring Boot JSP

- 在 main 路径下创建 webapp 文件夹，将所有的 JSP 资源放置在该文件夹下。
- 在 application.yml 添加视图解析相关配置。

```yaml
server:
  port: 9090
spring:
  mvc:
    view:
      prefix: /
      suffix: .jsp
```

TestHandler

```java
package com.southwind.controller;

import com.southwind.entity.Student;
import com.southwind.service.StudentService;
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.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
@RequestMapping("/test")
public class TestHandler {

    @Autowired
    private StudentService studentService;

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

    @GetMapping("/findById")
    public ModelAndView findById(long id){
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("update");
        modelAndView.addObject("student",studentService.findById(id));
        return modelAndView;
    }

    @PostMapping("/update")
    public String update(Student student){
        studentService.update(student);
        return "redirect:/test/findAll";
```

```
    }

    @PostMapping("/save")
    public String save(Student student){
        studentService.save(student);
        return "redirect:/test/findAll";
    }

    @GetMapping("/deleteById")
    public String deleteById(long id){
        studentService.deleteById(id);
        return "redirect:/test/findAll";
    }
}
```