笔记本：	2018太极第一周作业笔记		
创建时间：	2018/11/25 0:54	**更新时间：**	2018/11/26 10:04
作者：	2363655324idjzb		

Spring-boot-sample项目

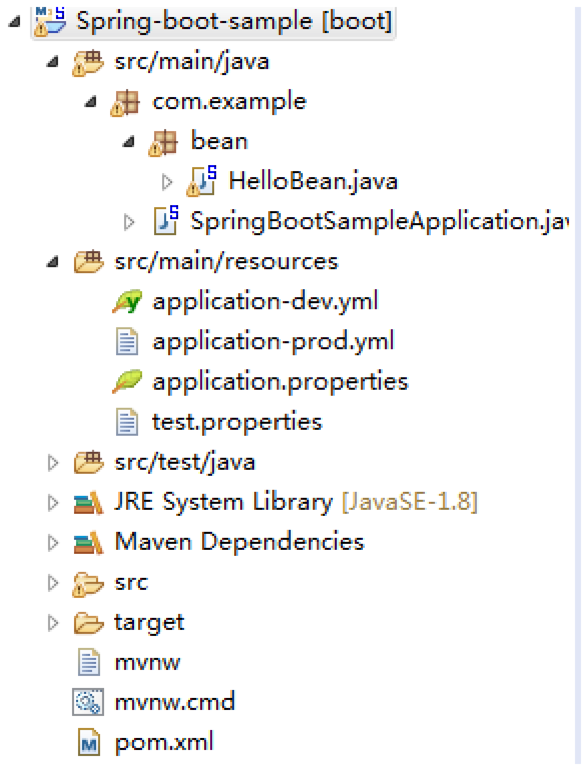
作者： 2363655324idjzb

Spring-boot-sample项目

0.功能

练习日志配置-logback和log4j2

1.项目结构



```
Spring-boot-sample [boot]
  src/main/java
    com.example
      bean
        HelloBean.java
      SpringBootSampleApplication.ja
  src/main/resources
    application-dev.yml
    application-prod.yml
    application.properties
    test.properties
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  mvnw
  mvnw.cmd
  pom.xml
```

2.在pom.文件中加入web依赖

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

3.代码实现

3.1 bean文件夹中的实体类HelloBean.java 文件

```
package com.example.bean;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Component;

import lombok.Data;
import lombok.ToString;

@ConfigurationProperties(prefix="my")
@Configuration
@Data

public class HelloBean {
    private String secret;
    private String number;
    private String bignumber;

}
```

3.2 SpringBootSampleApplication启动类文件

```
package com.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.bean.HelloBean;

@SpringBootApplication
@RestController
public class SpringBootSampleApplication {

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

    @RequestMapping("/hello")
    public String hello() {
        return "hello,world!"+name;
    }

    @Autowired
    private HelloBean helloBean;

    @RequestMapping("/hellobean")
    public String hellobean() {
        return helloBean.toString();

    }
    @Bean
    public static CommandLineRunner testA() {
        CommandLineRunner runner = new CommandLineRunner() {
            @Override
            public void run(String... args) throws Exception {
                System.out.println("The testA runner start to init...");

            }
        };
        return runner;
    }



    public static void main(String[] args) {

        SpringApplication.run(SpringBootSampleApplication.class, args);
    }
}
```

3.3 多配置文件的使用

applicatioon.properties文件

```
server.port=8082
spring.application.name=sample
name=jimmy
spring.profiles.active=dev
```

applicatioon-dev.properties文件

```
server:
  port: 8082



my:

   secret: wang

   number: xiao

   bignumber: wang
```

applicatioon-prod.properties文件

```
name: Tom
server:
  port: 8082
```

applicatioon-test.properties文件

```
com.example.source=1
com.example.age=12
```

Spring-boot-profiles项目

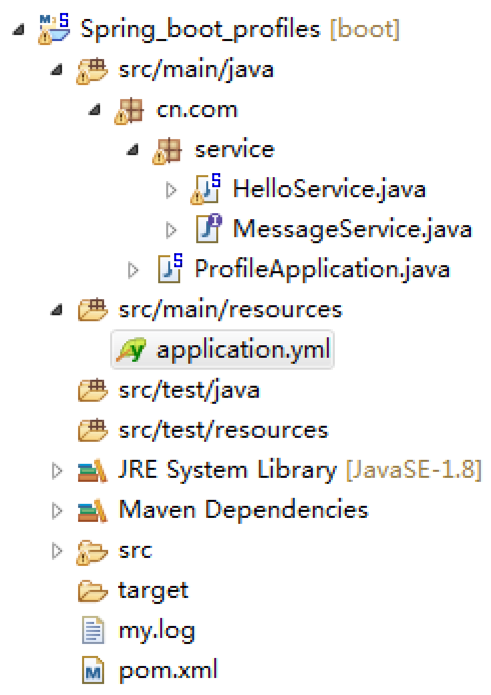
作者： 2363655324idjzb

Spring-boot-profiles项目

0.功能

综合练习

1.项目结构


```
Spring_boot_profiles [boot]
  src/main/java
    cn.com
      service
        HelloService.java
        MessageService.java
      ProfileApplication.java
  src/main/resources
    application.yml
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  my.log
  pom.xml
```

2.在pom.文件中加入依赖

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

3.代码实现

3.1 service文件夹中的实体类HelloService.java 文件

```
package cn.com.service;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Profile;
import org.springframework.stereotype.Component;

@Component
@Profile("hello")
public class HelloService implements MessageService {

    @Value("${name:world}")
    private String name;

    @Override
    public String message() {

        return "hello" + this.name;
    }

}
```

3.2service文件夹下的MessageService.java文件

```
packagecn.com.service;
public interface MessageService {
    public String message();
}
```

3.3 ProfileApplication启动类文件

```
package cn.com;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import cn.com.service.HelloService;

@SpringBootApplication
@RestController
public class ProfileApplication implements CommandLineRunner {

    @Autowired
    private HelloService hellowang;

    @RequestMapping("/helloService")
    public void hello() {
        System.out.println(hellowang.message());
    }

    // 命令行启动器
    public void run(String... args) throws Exception {
        System.out.println(this.hellowang.message());

    }

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

}
```

3.3 配置文件的使用

applicatioon.properties文件

```
server:
  port: 8001

management:

  port: 8889

  context-path: /abc

  security:

    enabled: false

---

logging:

  file: my.log

  level:

    root: warn

    org:

      springframework:

        web: debug

---

spring:

  profiles:

    active: hello


---

spring:

  profiles: hello

---

spring:

  profiles: goodbye
```

MySpring项目

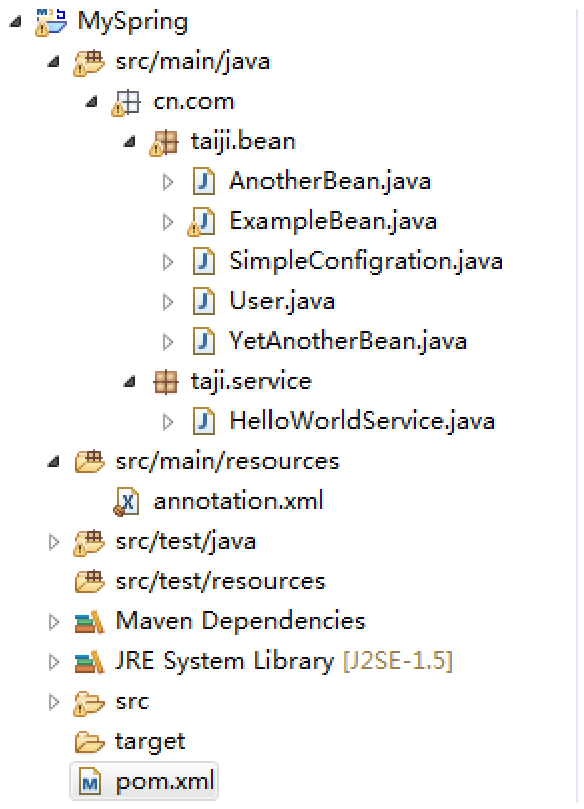
作者： 2363655324idjzb

MySpring项目

0.功能

练习依赖注入

1.项目结构


MySpring
src/main/java
cn.com
taiji.bean
AnotherBean.java
ExampleBean.java
SimpleConfigration.java
User.java
YetAnotherBean.java
taji.service
HelloWorldService.java
src/main/resources
annotation.xml
src/test/java
src/test/resources
Maven Dependencies
JRE System Library [J2SE-1.5]
src
target
pom.xml

2.在pom.文件中加入web依赖

```
<dependencyManagement>
        <dependencies>

            <dependency>

                <groupId>io.spring.platform</groupId>
                <artifactId>platform-bom</artifactId>
                <version>Cairo-SR5</version>

                <type>pom</type>

                <scope>import</scope>

            </dependency>

        </dependencies>

</dependencyManagement>

<dependencies>

        <dependency>

            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </dependency>

        <dependency>

            <groupId>javax.inject</groupId>

            <artifactId>javax.inject</artifactId>
        </dependency>

        <dependency>

            <groupId>junit</groupId>

            <artifactId>junit</artifactId>
            <scope>test</scope>
        </dependency>
</dependencies>
```

3.代码实现

3.1 bean文件夹中的实体类User.java 文件

```
packagecn.com.taiji.bean;
importorg.springframework.stereotype.Component;
@Component
public class User {
    private String name;
    private int age;
    public User() {
        super();
    }
    public User(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "User [name=" + name + ", age=" + age + "]";
    }
}
```

3.2 bean文件夹中的实体类AnotherBean.java 文件

```
packagecn.com.taiji.bean;
importorg.springframework.stereotype.Component;
```

```java
@Component
public class AnotherBean {
    private String name;

    private ExampleBean exampleBean;
    public String getName() {
        return name;
    }
    public AnotherBean() {
        super();
    }
    public AnotherBean(String name, ExampleBean exampleBean) {
        super();
        this.name = name;
        this.exampleBean = exampleBean;
    }
    public void setName(String name) {
        this.name = name;
    }
    public ExampleBean getExampleBean() {
        return exampleBean;
    }
    public void setExampleBean(ExampleBean exampleBean) {
        this.exampleBean = exampleBean;
    }
}
```

3.3 bean文件夹中的实体类ExampleBean.java 文件

```java
package  cn.com.taiji.bean;

import javax.annotation.Resource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import  org.springframework.stereotype.Component;

@Component
public class ExampleBean {
    @Autowired
    // @Qualifier("ab")
    // @Resource(name = "ab")
    private AnotherBean beanOne;

    @Autowired
    private YetAnotherBean beanTwo;

    private int i;

    public ExampleBean() {
        super();
    }

    public ExampleBean(AnotherBean beanOne, YetAnotherBean beanTwo, int i) {
        super();
        this.beanOne = beanOne;
        this.beanTwo = beanTwo;
        this.i = i;
    }

    public AnotherBean getBeanOne() {
        return beanOne;
    }

    public void setBeanOne(AnotherBean beanOne) {
        this.beanOne = beanOne;
    }

    public YetAnotherBean getBeanTwo() {
        return beanTwo;
    }

    public void setBeanTwo(YetAnotherBean beanTwo) {
        this.beanTwo = beanTwo;
    }

    public int getI() {
        return i;
    }

    public void setI(int i) {
        this.i = i;
    }
}
```

3.4 bean文件夹中的实体类SimpleConfigration.java 文件

```
package  cn.com.taiji.bean;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;

@Configuration
public class SimpleConfigration {

    @Bean
    @Primary
    public User user() {
        User user = new User();
        user.setAge(10);
        user.setName("tom");
        return user;
    }
}
```

3.5 service文件夹中的实体类HelloWorldService.java 文件

```
package  cn.com.taji.service;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

@Service
public class HelloWorldService {
    @Value("aaa")
    private String name;

    public HelloWorldService() {

    }

    public HelloWorldService(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

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

    public void sayHello() {
        System.out.println("hello" + this.name);
    }
}
```

Spring-boot-one-3项目

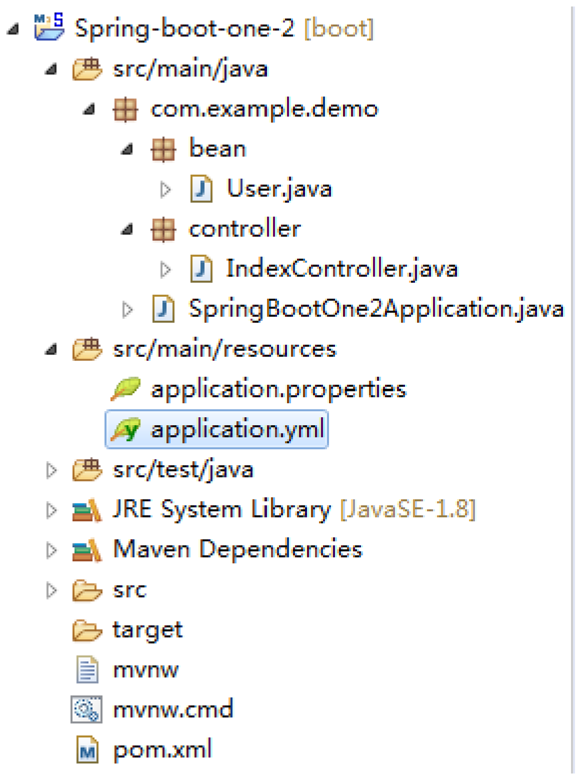
作者： 2363655324idjzb

Spring-boot-one-3项目

0.功能

练习使用多配置文件

1.项目结构



```
Spring-boot-one-2 [boot]
  src/main/java
    com.example.demo
      bean
        User.java
      controller
        IndexController.java
      SpringBootOne2Application.java
  src/main/resources
    application.properties
    application.yml
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  mvnw
  mvnw.cmd
  pom.xml
```

2.在pom.文件中加入web依赖

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

3.代码实现

3.1 bean文件夹中的实体类User.java 文件

```
package com.example.demo.bean;
import java.util.Date;
public class User {
    private int id;
    private String name;
    private Date date;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
```

```java
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
}
```

3.2 controller文件夹下的IndexController.java文件

```java
package com.example.demo.controller;
import java.util.Date;
import java.util.HashMap;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.bean.User;
@RestController
@RequestMapping(value = "/index")
public class IndexController {
    @RequestMapping
    public String index() {
        return "hello world";
    }
    // @RequestParam 简单类型的绑定，可以出来get和post
    @RequestMapping(value = "/get")
    public HashMap<String, Object> get(@RequestParam String name) {
        HashMap<String, Object> map = new HashMap<String, Object>();
        map.put("title", "hello world");
        map.put("name", name);
        return map;
    }
    // @PathVariable 获得请求url中的动态参数
    @RequestMapping(value = "/get/{id}/{name}")
    public User getUser(@PathVariable int id, @PathVariable String name) {
        User user = new User();
        user.setId(id);
        user.setName(name);
        user.setDate(new Date());
        return user;
    }
}
```

3.3 SpringBootOne2Application启动类文件

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringBootOne1Application {
    public static void main(String[] args) {
        SpringApplication.run(SpringBootOne1Application.class, args);
    }
}
```

3.3 多配置文件的使用

applicatioon.properties文件

```
spring.profiles.active=dev
```

applicatioon-dev.properties文件

```
server.port=8080
```

applicatioon-prod.properties文件

```
server.port=8081
```

applicatioon-test.properties文件

```
server.port=8083
```

application.yml文件

```
#配置文件环境配置
spring:
  profiles:
    active: dev
#端口
server:
  port: 8888

---
spring:
  profiles: dev
```

```
server:
  port: 8080

---
spring:
  profiles: prod
server:
  port: 8082

---
spring:
  profiles: test
server:
  port: 8081
```

Spring-boot-one-4项目

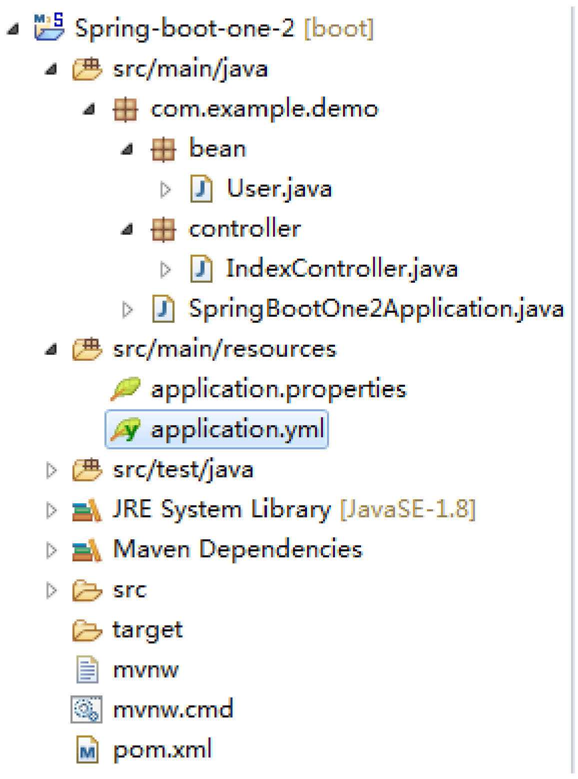
作者： 2363655324idjzb

Spring-boot-one-4项目

0.功能

练习日志配置-logback和log4j2

1.项目结构



```
Spring-boot-one-2 [boot]
  src/main/java
    com.example.demo
      bean
        User.java
      controller
        IndexController.java
      SpringBootOne2Application.java
  src/main/resources
    application.properties
    application.yml
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  mvnw
  mvnw.cmd
  pom.xml
```

2.在pom.文件中加入web依赖

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

3.代码实现

3.1 bean文件夹中的实体类User.java 文件

```
package com.example.demo.bean;
import java.util.Date;
public class User {
    private int id;
    private String name;
    private Date date;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
}
```

3.2 controller文件夹下的IndexController.java文件

```
package com.example.demo.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.Date;
import java.util.HashMap;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.bean.User;
@RestController
@RequestMapping(value = "/index")
public class IndexController {
    private static final Logger logger =
LoggerFactory.getLogger(IndexController.class);
    @RequestMapping
    public String index() {
        return "hello world";
    }



    @RequestMapping
    public String index() {
        logger.debug("this is a log test, debug");
        logger.info("this is a log test, info");
        return "hello world";
    }
    // @RequestParam 简单类型的绑定，可以出来get和post
    @RequestMapping(value = "/get")
    public HashMap<String, Object> get(@RequestParam String name) {
        HashMap<String, Object> map = new HashMap<String, Object>();
        map.put("title", "hello world");
        map.put("name", name);
        return map;
    }
    // @PathVariable 获得请求url中的动态参数
    @RequestMapping(value = "/get/{id}/{name}")
    public User getUser(@PathVariable int id, @PathVariable String name) {
        User user = new User();
        user.setId(id);
        user.setName(name);
        user.setDate(new Date());
        return user;
    }
}
```

3.3 SpringBootOne2Application启动类文件

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringBootOne1Application {
    public static void main(String[] args) {
        SpringApplication.run(SpringBootOne1Application.class, args);
    }
}
```

3.3 多配置文件的使用

applicatioon.properties文件

```
spring.profiles.active=dev

logging.config=classpath:logback-wang.xml

# 应用自定义配置
#logging.config=classpath:log4j2-dev.xml
```

applicatioon-dev.properties文件

```
server.port=8080
```

applicatioon-prod.properties文件

```
server.port=8081
```

applicatioon-test.properties文件

```
server.port=8083
```

logback-wang.xml文件

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

    <!-- 文件输出格式 -->
    <property name="PATTERN" value="%-12(%d{yyyy-MM-dd HH:mm:ss.SSS}) |-%-5level
[%thread] %c [%L] -| %msg%n" />
    <!-- test文件路径 -->
    <property name="TEST_FILE_PATH" value="c:/wang/logs" />

    <!-- pro文件路径 -->
    <property name="PRO_FILE_PATH" value="/wang/logs" />

</configuration>
```

log4j2-dev.xml文件

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
   <properties>
      <!-- 文件输出格式 -->
      <property name="PATTERN">%d{yyyy-MM-dd HH:mm:ss.SSS} |-%-5level [%thread] %c [%L] -|
%msg%n</property>
   </properties>

   <appenders>
      <Console name="CONSOLE" target="system_out">
         <PatternLayout pattern="${PATTERN}" />
      </Console>
   </appenders>

   <loggers>
      <logger name="com.wang.com" level="debug" />
      <root level="info">
         <appenderref ref="CONSOLE" />
      </root>
   </loggers>

</configuration>
```

Spring-boot-one-1项目

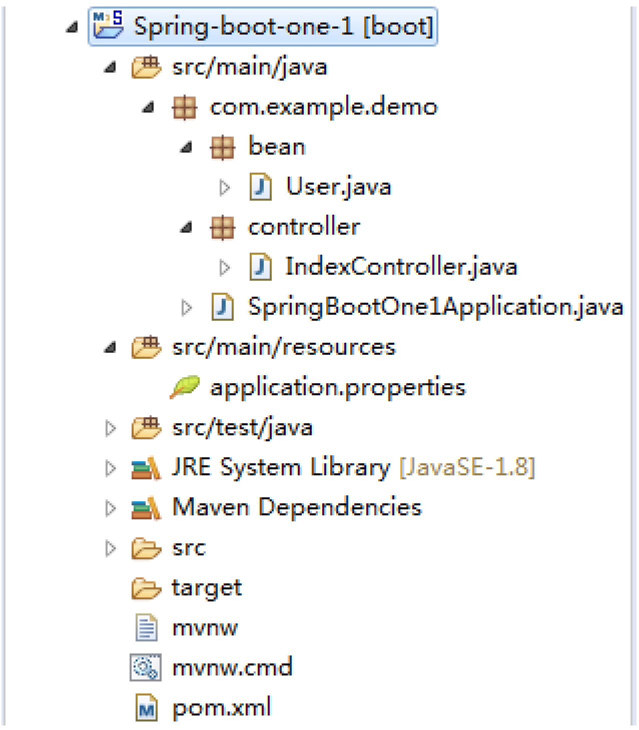
作者： 2363655324idjzb

Spring-boot-one-1项目

0.功能

简单的项目的搭建运行，web网页上获取User的一些数据

1.项目结构



```
Spring-boot-one-1 [boot]
  src/main/java
    com.example.demo
      bean
        User.java
      controller
        IndexController.java
      SpringBootOne1Application.java
  src/main/resources
    application.properties
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  mvnw
  mvnw.cmd
  pom.xml
```

2.在pom.文件中加入web依赖

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

3.代码实现

3.1 bean文件夹中的实体类User.java 文件

```
package com.example.demo.bean;
import java.util.Date;
public class User {
    private int id;
    private String name;
    private Date date;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```java
        this.name = name;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
}
```

3.2 controller文件夹下的IndexController.java文件

```java
package com.example.demo.controller;
import java.util.Date;
import java.util.HashMap;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.bean.User;
@RestController
@RequestMapping(value = "/index")
public class IndexController {
    @RequestMapping
    public String index() {
        return "hello world";
    }
    // @RequestParam 简单类型的绑定，可以出来get和post
    @RequestMapping(value = "/get")
    public HashMap<String, Object> get(@RequestParam String name) {
        HashMap<String, Object> map = new HashMap<String, Object>();
        map.put("title", "hello world");
        map.put("name", name);
        return map;
    }
    // @PathVariable 获得请求url中的动态参数
    @RequestMapping(value = "/get/{id}/{name}")
    public User getUser(@PathVariable int id, @PathVariable String name) {
        User user = new User();
        user.setId(id);
        user.setName(name);
        user.setDate(new Date());
        return user;
    }
}
```

3.3 SpringBootOne1Application启动类文件

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringBootOne1Application {
    public static void main(String[] args) {
        SpringApplication.run(SpringBootOne1Application.class, args);
    }
}
```

4.直接运行main方法或者使用maven命令:

```
mvn spring-boot:run
```

5.打包命令:

```
clean package
```

6.运行命令:

```
java  - jar com.example.spring-boot-one-1-0.0.1-SNAPSHOT.jar
```

Spring-boot-one-2项目

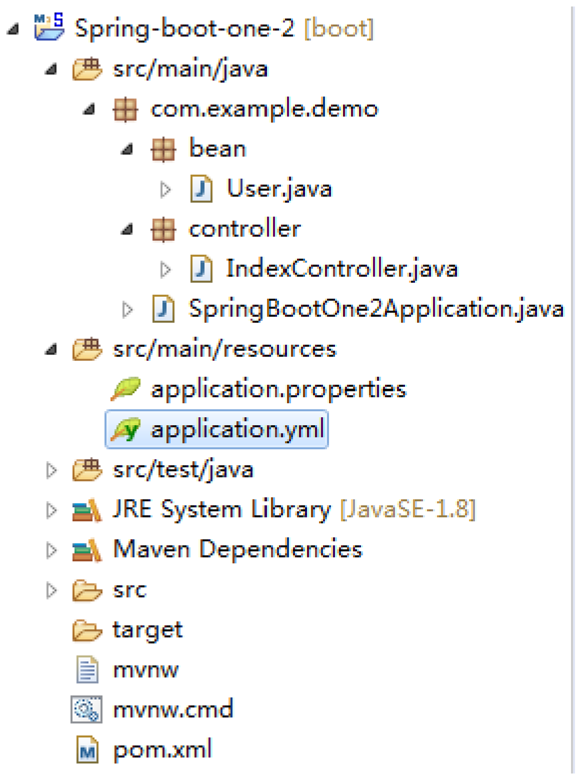
作者： 2363655324idjzb

Spring-boot-one-2项目

0.功能

练习使用配置文件

1.项目结构



```
Spring-boot-one-2 [boot]
  src/main/java
    com.example.demo
      bean
        User.java
      controller
        IndexController.java
      SpringBootOne2Application.java
  src/main/resources
    application.properties
    application.yml
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  mvnw
  mvnw.cmd
  pom.xml
```

2.在pom.文件中加入web依赖

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

3.代码实现

3.1 bean文件夹中的实体类User.java 文件

```
package com.example.demo.bean;
import java.util.Date;
public class User {
    private int id;
    private String name;
    private Date date;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
}
```

3.2 controller文件夹下的IndexController.java文件

```
package com.example.demo.controller;
import java.util.Date;
import java.util.HashMap;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.bean.User;
@RestController
@RequestMapping(value = "/index")
public class IndexController {

    @Value(value = "${my.secret}")
    private String secret;

    @Value(value = "${my.number}")
    private int id;

    @Value(value = "${my.desc}")
    private String desc;
    @RequestMapping
    public String index() {
        return "hello world";
    }
    // @RequestParam 简单类型的绑定，可以出来get和post
    @RequestMapping(value = "/get")
    public HashMap<String, Object> get(@RequestParam String name) {
        HashMap<String, Object> map = new HashMap<String, Object>();
        map.put("title", "hello world");
        map.put("name", name);
        map.put("secret", secret);
        map.put("id", id);
        map.put("desc", desc);
        return map;
    }
    // @PathVariable 获得请求url中的动态参数
    @RequestMapping(value = "/get/{id}/{name}")
    public User getUser(@PathVariable int id, @PathVariable String name) {
        User user = new User();
        user.setId(id);
        user.setName(name);
        user.setDate(new Date());
        return user;
    }
}
```

3.3 SpringBootOne2Application启动类文件

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringBootOne1Application {
    public static void main(String[] args) {
        SpringApplication.run(SpringBootOne1Application.class, args);
    }
}
```

3.3 配置文件的使用

applicatioon.properties文件

```
my.secret=${random.value}
my.number=${random.int}
my.name=www.wangxiaowang.com
my.desc=the domain is ${my.name}
server.port=8080
```

application.yml文件

```
#自定义配置
my:
  secret: ${random.value}
  number: ${random.int}
  name:www.wangwang.com
  desc: the domain is ${my.name}

#端口
server:
  port: 9090
```