# learnSpringBoot

一、热身

Spring boot 最小化配置，微服务。

看了一些关于Spring boot的书和文章，觉得有必要写个小实例来检验学习。

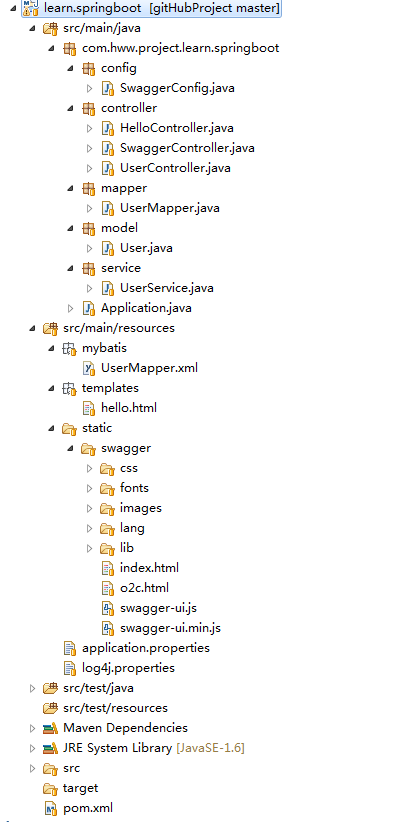
二、实践

一些说明：

工具使用Eclispe

工程依赖管理采用个人比较熟悉的Maven

（2）完整项目结构



（3）数据库

数据库名：test

【user.sql】

SET FOREIGN\_KEY\_CHECKS=0;

-- ----------------------------

-- Table structure for user

-- ----------------------------

DROP TABLE IF EXISTS `user`;

CREATE TABLE `user` (

`id` int(11) NOT NULL,

`name` varchar(255) DEFAULT NULL,

`age` int(11) DEFAULT NULL,

`password` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- ----------------------------

-- Records of user

-- ----------------------------

INSERT INTO `user` VALUES ('1', '7player', '18', '123456');

2.Maven配置

完整的【pom.xml】配置如下：

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

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.hww.project</groupId>

<artifactId>learn.springboot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.0.RELEASE</version>

</parent>

<!-- Add typical dependencies for a web application -->

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<!--模板引擎-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<!--支持使用 JDBC 访问数据库-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<!--Mybatis-->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.2.2</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.2.8</version>

</dependency>

<!--Mysql / DataSource-->

<dependency>

<groupId>org.apache.tomcat</groupId>

<artifactId>tomcat-jdbc</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<!--Json Support-->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.1.43</version>

</dependency>

<!--Swagger support，Swagger是当前最好用的Restful API文档生成的开源项目-->

<dependency>

<groupId>com.mangofactory</groupId>

<artifactId>swagger-springmvc</artifactId>

<version>0.9.5</version>

</dependency>

</dependencies>

<repositories>

<repository>

<id>spring-snapshots</id>

<url>http://repo.spring.io/libs-snapshot</url>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

<id>spring-snapshots</id>

<url>http://repo.spring.io/libs-snapshot</url>

</pluginRepository>

</pluginRepositories>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

3.主函数

【Application.java】包含main函数，像普通java程序启动即可。

此外，该类中还包含和数据库相关的DataSource，SqlSeesion配置内容。

注：@MapperScan(“cn.no7player.mapper”) 表示Mybatis的扫描的mapper映射包路径

package com.hww.project.learn.springboot;

import org.apache.ibatis.session.SqlSessionFactory;

import org.apache.log4j.Logger;

import org.apache.tomcat.jdbc.pool.DataSource;

import org.mybatis.spring.SqlSessionFactoryBean;

import org.mybatis.spring.annotation.MapperScan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.core.io.support.PathMatchingResourcePatternResolver;

import org.springframework.jdbc.datasource.DataSourceTransactionManager;

import org.springframework.transaction.PlatformTransactionManager;

@EnableAutoConfiguration

@SpringBootApplication

@ComponentScan

@MapperScan("com.hww.project.learn. springboot.mapper")

public class Application {

private static Logger logger = Logger.getLogger(Application.class);

//DataSource配置

@Bean

@ConfigurationProperties(prefix="spring.datasource")

public DataSource dataSource() {

return new org.apache.tomcat.jdbc.pool.DataSource();

}

//提供SqlSeesion

@Bean

public SqlSessionFactory sqlSessionFactoryBean() throws Exception {

SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();

sqlSessionFactoryBean.setDataSource(dataSource());

PathMatchingResourcePatternResolver resolver = new PathMatchingResourcePatternResolver();

sqlSessionFactoryBean.setMapperLocations(resolver.getResources("classpath:/mybatis/\*.xml"));

return sqlSessionFactoryBean.getObject();

}

@Bean

public PlatformTransactionManager transactionManager() {

return new DataSourceTransactionManager(dataSource());

}

/\*\*

\* Main Start

\*/

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

logger.info("============= SpringBoot Start Success =============");

}

}

4.Controller

请求入口Controller部分提供三种接口样例：视图模板，Json，restful风格

（1）视图模板

返回结果为视图文件路径。视图相关文件默认放置在路径 resource/templates下：

package com.hww.project.learn.springboot.controller;

import org.apache.log4j.Logger;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

import org.springframework.web.bind.annotation.RequestParam;

@Controller

public class HelloController {

private Logger logger = Logger.getLogger(HelloController.class);

/\*

\* http://localhost:8080/hello?name=cn.7player

\*/

@RequestMapping("/hello")

public String greeting(@RequestParam(value="name", required=false, defaultValue="World") String name, Model model) {

logger.info("hello");

model.addAttribute("name", name);

return "hello";

}

}（2）Json

返回Json格式数据，多用于Ajax请求。

package com.hww.project.learn.springboot.controller;

import org.apache.log4j.Logger;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

import org.springframework.web.bind.annotation.ResponseBody;

import com.hww.project.learn.springboot.model.User;

import com.hww.project.learn.springboot.service.UserService;

@Controller

public class UserController {

private Logger logger = Logger.getLogger(UserController.class);

@Autowired

private UserService userService;

/\*

\* http://localhost:8080/getUserInfo

\*/

@RequestMapping("/getUserInfo")

@ResponseBody

public User getUserInfo() {

User user = userService.getUserInfo();

if(user!=null){

System.out.println("user.getName():"+user.getName());

logger.info("user.getAge():"+user.getAge());

}

return user;

}

}

（3）restful

REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。

此外，有一款RESTFUL接口的文档在线自动生成+功能测试功能软件——Swagger UI，具体配置过程可移步《Spring Boot 利用 Swagger 实现restful测试》

package com.hww.project.learn.springboot.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PathVariable;

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

import org.springframework.web.bind.annotation.RequestMethod;

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

import com.hww.project.learn.springboot.model.User;

import com.hww.project.learn.springboot.service.UserService;

import com.wordnik.swagger.annotations.ApiOperation;

@RestController

@RequestMapping(value="/users")

public class SwaggerController {

/\*

\* http://localhost:8080/swagger/index.html

\*/

@Autowired

private UserService userService;

@ApiOperation(value="Get all users",notes="requires noting")

@RequestMapping(method=RequestMethod.GET)

public List<User> getUsers(){

return userService.getAllUser();

}

@ApiOperation(value="Get user with id",notes="requires the id of user")

@RequestMapping(value="/{name}",method=RequestMethod.GET)

public User getUserByName(@PathVariable String name){

return userService.getUserByName(name);

}

}

5.Mybatis

配置相关代码在Application.java中体现。

（1）【application.properties】

spring.datasource.url=jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=gbk&zeroDateTimeBehavior=convertToNull

spring.datasource.username=root

spring.datasource.password=123456

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

注意，在Application.java代码中，配置DataSource时的注解

@ConfigurationProperties(prefix=“spring.datasource”)

表示将根据前缀“spring.datasource”从application.properties中匹配相关属性值。

（2）【UserMapper.xml】

Mybatis的sql映射文件。Mybatis同样支持注解方式，在此不予举例了。

<?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.hww.project.learn.springboot.mapper.UserMapper"*>

<select id=*"findUserInfo"* resultType=*"com.hww.project.learn.springboot.model.User"*>

select id,name, age,password from user

</select>

<select id=*"getAllUser"* resultType=*"com.hww.project.learn.springboot.model.User"*>

select id,name, age,password from user

</select>

<select id=*"getUserByName"* resultType=*"com.hww.project.learn.springboot.model.User"*

parameterType=*"string"*>

select id,name, age,password from user

where name = #{name}

</select>

</mapper>

（3）接口UserMapper

package com.hww.project.learn.springboot.mapper;

import java.util.List;

import com.hww.project.learn.springboot.model.User;

public interface UserMapper {

public User findUserInfo();

public List<User> getAllUser();

public User getUserByName(String name);

}

（4）service类

package com.hww.project.learn.springboot.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.hww.project.learn.springboot.mapper.UserMapper;

import com.hww.project.learn.springboot.model.User;

@Service

public class UserService {

@Autowired

private UserMapper userMapper;

public User getUserInfo() {

return userMapper.findUserInfo();

}

public List<User> getAllUser(){

return userMapper.getAllUser();

}

public User getUserByName(String name){

return userMapper.getUserByName(name);

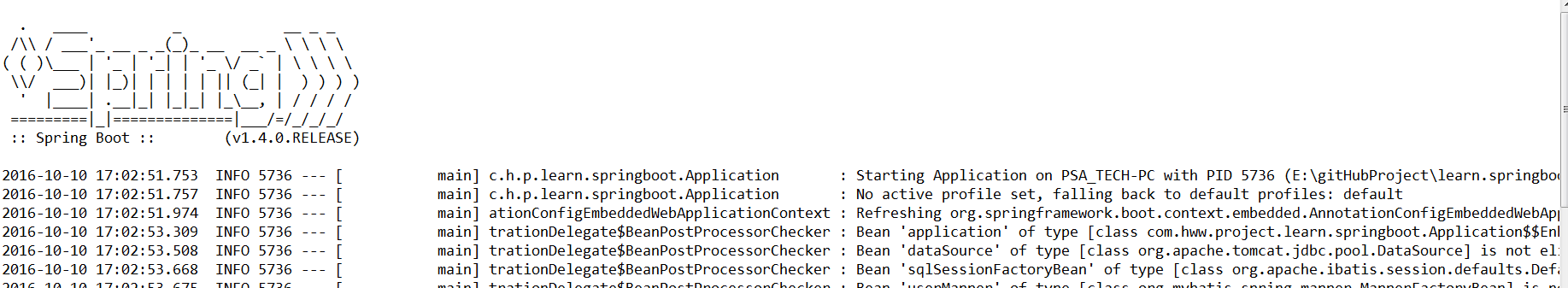
}

}

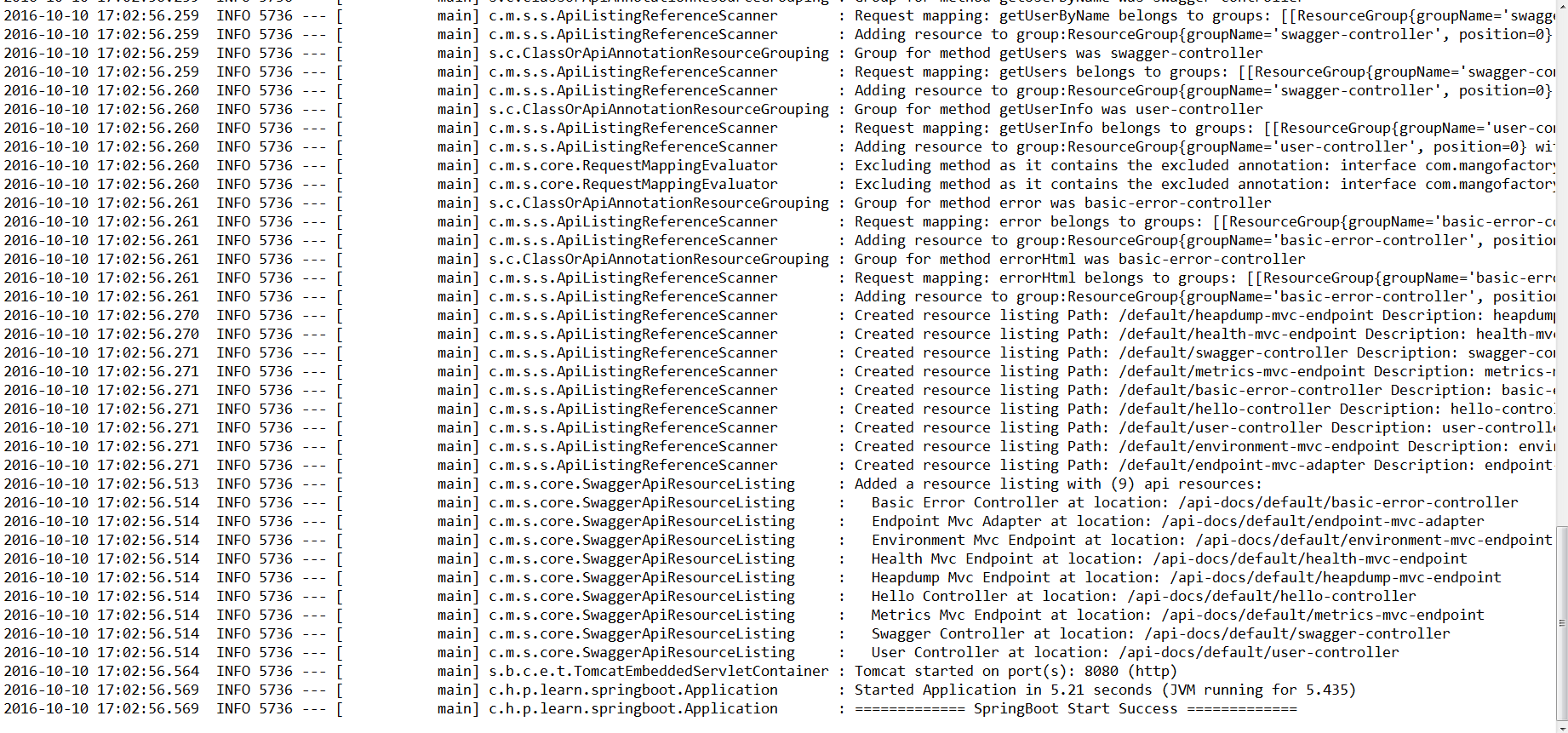
三、总结

（1）运行 Application.java

（2）控制台输出：



…..（略过无数内容）



（3）访问：

针对三种控制器的访问分别为：

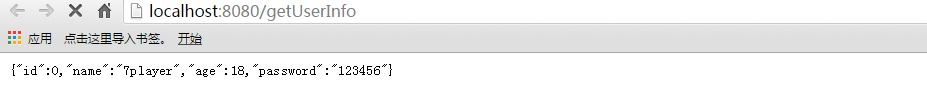
视图：

http://localhost:8080/hello?name=huanggai1



Json：

http://localhost:8080/getUserInfo



Restful（使用了swagger）：

http://localhost:8080/swagger/index.html

