springboot和swagger2整合

1、swagger2介绍

wagger 是一款RESTFUL接口的文档在线自动生成+功能测试功能软件。

Swagger 是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务。总体目标是使客户端和文件系统作为服务器以同样的速度来更新。文件的方法，参数和模型紧密集成到服务器端的代码，允许API来始终保持同步。Swagger 让部署管理和使用功能强大的API从未如此简单。

官网：http://swagger.io/

GitHub地址：https://github.com/swagger-api/swagger-ui

2、环境搭建

2.1 创建springboot的maven项目

略

2.2 添加依赖

```
<!--依赖管理父节点的配置，简化maven的依赖管理-->
<parent>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-parent</artifactId>
   <version>1.5.10.RELEASE</version>
   <relativePath/> <!-- lookup parent from repository -->
</parent>



<!--配置启动依赖和其他依赖-->
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>

   <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>5.1.28</version>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
```

```
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
    </dependency>

    <!--swagger2 依赖-->
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger-ui</artifactId>
        <version>2.7.0</version>
    </dependency>

    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger2</artifactId>
        <version>2.7.0</version>
    </dependency>

</dependencies>

<!--配置maven插件：可以用maven的形式启动Spring Boot项目（另外也可以在主方法中启动）-->
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

2.3 springboot的配置文件

application.properties

```
#修改tomcat启动端口号
server.port=8088
#修改项目的部署路径
#server.servlet.context-path=/book
server.context-path=/book
```

```
#修改日志级别
logging.level.root=ERROR


#mysql配置
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=admin
spring.datasource.url=jdbc:mysql://localhost:3306/ssh?
useUnicode=true&characterEncoding=UTF-8

#jp配置
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

2.4 创建swagger2的配置类

```
package com.itqf;

import com.google.common.base.Predicates;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

/**
 * Created by liliting on 18/3/14.
 */
@Configuration
@EnableSwagger2
public class SwaggerConfig
{

    @Bean
    public Docket getDocket(){

        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .useDefaultResponseMessages(false)
                .select()
                //.paths(Predicates.or(PathSelectors.regex("/api2/.*")))//  过滤
该路径下的请求生成RESTful api
                .build();
    }

    private ApiInfo apiInfo(){
        return new ApiInfoBuilder()
                .title("BookStore Platform API")//大标题
                .description("BookStore Platform's REST API, all the applications
could access the Object model data via JSON.")//详细描述
                .version("2.0")//版本
                .contact(new Contact("java", "http://itqf.com",
"123456@qq.com"))//作者
                .license("The Apache License, Version 2.0")//许可证信息
                .licenseUrl("http://www.apache.org/licenses/LICENSE-2.0.html")//许
可证地址
                .build();

    }

}
```

2.5 pojo类

```
package com.itqf.pojo;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.util.Date;

/**
 * Created by liliting on 18/3/14.
 */
@Entity
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private String author;

    private double price;

    private Date publishDate;

    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 String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public Date getPublishDate() {
        return publishDate;
    }

    public void setPublishDate(Date publishDate) {
        this.publishDate = publishDate;
    }
}
```

2.6 dao层

```
package com.itqf.dao.impl;

import com.itqf.pojo.Book;
import org.springframework.data.jpa.repository.JpaRepository;

/**
 * Created by liliting on 18/3/14.
 */
public  interface BookDaoImpl extends JpaRepository<Book,Integer> {


}
```

2.7 service层

2.7.1 接口

```
package com.itqf.service;

import com.itqf.pojo.Book;

import java.util.List;
import java.util.Optional;

/**
 * Created by liliting on 18/3/14.
 */
public interface BookService {

    public List<Book> findAll();

    public Book findById(int id);

    public void saveBook(Book book);

    public void update(Book book);

    public void delete(int id);
}
```

2.7.2 实现类

```
package com.itqf.service.impl;

import com.itqf.dao.impl.BookDaoImpl;
import com.itqf.pojo.Book;
import com.itqf.service.BookService;
import org.springframework.stereotype.Service;

import javax.annotation.Resource;
import java.util.List;
import java.util.Optional;

/**
 * Created by liliting on 18/3/14.
 */
@Service
public class BookServiceImpl  implements BookService {

    @Resource
    private BookDaoImpl bookDaoImpl;

    @Override
    public List<Book> findAll() {
        return bookDaoImpl.findAll();
    }

    @Override
    public Book  findById(int id) {
        return bookDaoImpl.findOne(id);

    }

    @Override
    public void saveBook(Book book) {
        bookDaoImpl.save(book);
    }

    @Override
    public void update(Book book) {
        bookDaoImpl.saveAndFlush(book);
    }

    @Override
    public void delete(int id) {
       bookDaoImpl.delete(id);
    }
}
```

2.8 编写Controller

```
package com.itqf.controller;

import com.itqf.pojo.Book;
import com.itqf.service.BookService;
import org.springframework.web.bind.annotation.*;

import javax.annotation.Resource;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Created by liliting on 18/3/14.
 */
//@Controller
@RestController   //Controller+ResponseBody
@RequestMapping("/api1")

public class BookController {
    @Resource
    private BookService bookService;


    @RequestMapping("/findAll")
    public List<Book> findAll()
    {
        System.out.print(111);
        return  bookService.findAll();
    }

    @PostMapping("/saveBook")
    public Book  save(@RequestBody  Book book){//json --》 对象
        bookService.saveBook(book);
        System.out.println(book);
        return book;
    }

    @PutMapping("/updateBook")
    public Book  update(Book book){
         bookService.update(book);
        return book;
    }

    @DeleteMapping("/deleteBook")
    public Map  delete(int id){
        Map map =  new HashMap();
      try{
          bookService.delete(id);
          map.put("result","success");
      }catch (Exception e){

          map.put("result","fail");
      }
        return map;
    }





}
```

2.9 启动类

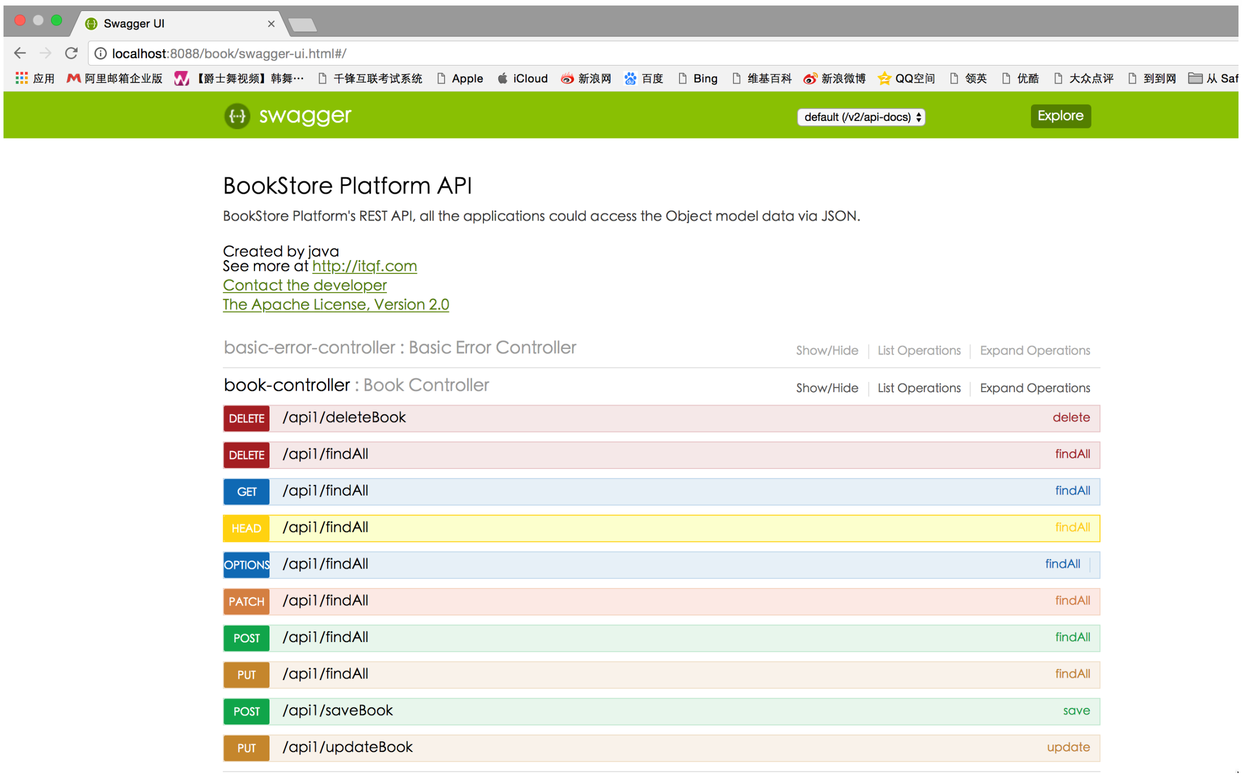
```
package com.itqf;

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

@SpringBootApplication
//位置：@SpringBootApplication
//1,main方法上  （常用）
//2,放到controller的类上 必须添加扫描其他包下的注解的代码 如： @ComponentScan("")
//SpringApplication.run 方法传的第一个参数的字节码对象就是Controller的class对象
public class SpringbootzhjpaApplication {

   public static void main(String[] args) {

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

3、访问swagger2主页

执行启动类，后方法如下地址：

http://localhost:8088/book/swagger-ui.html

效果如下：



BookStore Platform API

BookStore Platform's REST API, all the applications could access the Object model data via JSON.

Created by java
See more at http://itqf.com
Contact the developer
The Apache License, Version 2.0

basic-error-controller : Basic Error Controller — Show/Hide | List Operations | Expand Operations

book-controller : Book Controller — Show/Hide | List Operations | Expand Operations

DELETE	/api1/deleteBook	delete
DELETE	/api1/findAll	findAll
GET	/api1/findAll	findAll
HEAD	/api1/findAll	findAll
OPTIONS	/api1/findAll	findAll
PATCH	/api1/findAll	findAll
POST	/api1/findAll	findAll
PUT	/api1/findAll	findAll
POST	/api1/saveBook	save
PUT	/api1/updateBook	update

4、添加swagger文档注解

swagger默认生成的api文档都是英文的，可添加swagger文档注解，使api可得更通俗易懂。

常用注解：

- **@Api()**用于类；
  表示标识这个类是swagger的资源
- **@ApiOperation()**用于方法；
  表示一个http请求的操作
- **@ApiParam()**用于方法，参数，字段说明；
  表示对参数的添加元数据（说明或是否必填等）
- **@ApiModel()**用于类
  表示对类进行说明，用于参数用实体类接收
- **@ApiModelProperty()**用于方法，字段
  表示对model属性的说明或者数据操作更改
- **@ApiIgnore()**用于类，方法，方法参数
  表示这个方法或者类被忽略
- **@ApiImplicitParam()** 用于方法
  表示单独的请求参数
- **@ApiImplicitParams()** 用于方法，包含多个 @ApiImplicitParam

在controller中和pojo中的添加注解：

如下

4.1 pojo

```
package com.itqf.pojo;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.util.Date;

/**
 * Created by liliting on 18/3/14.
 */
@Entity
@ApiModel(value = "图书对象",description = "对应图书表")
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @ApiModelProperty(name="id",example = "1")
    private int id;
    @ApiModelProperty(name="name",example = "天龙八部")
    private String name;
    @ApiModelProperty(name="author",example = "金庸")
    private String author;
    @ApiModelProperty(name="price",example = "250.0")

    private double price;
    @ApiModelProperty(name="publishDate",example = "2018-03-20")

    private Date publishDate;

    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;
    }
```

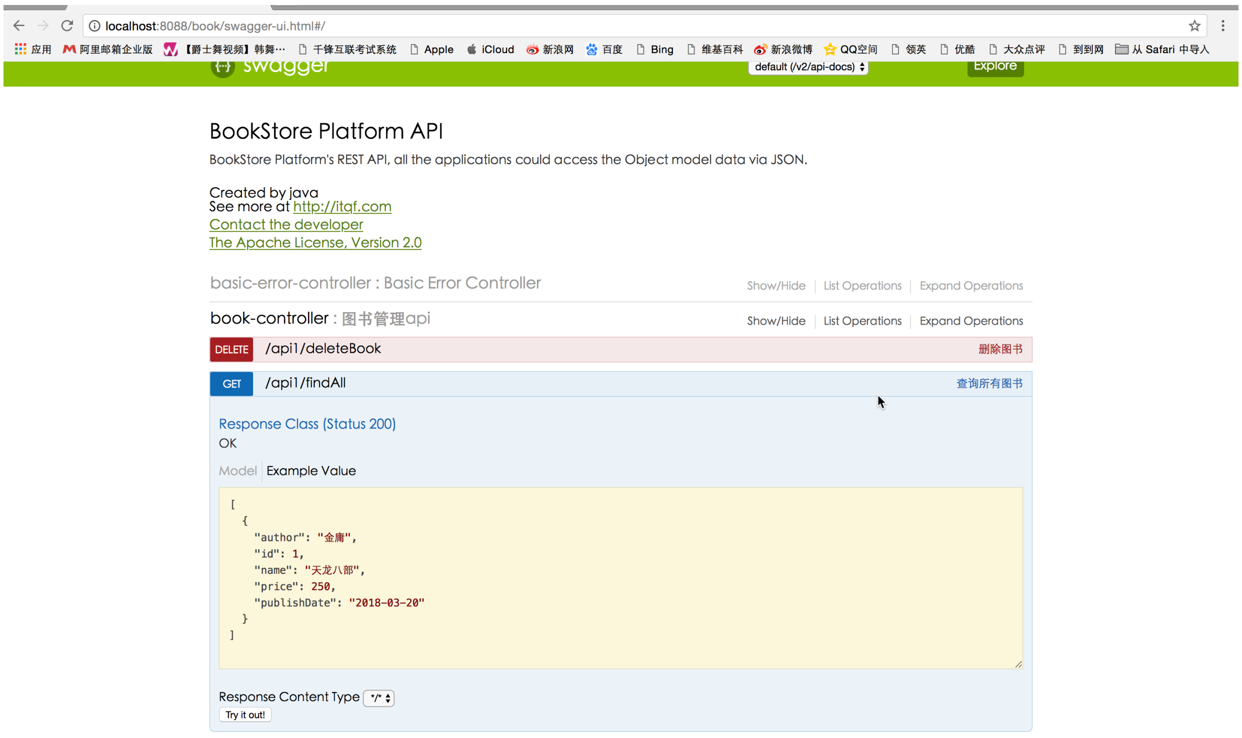
```java
    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public Date getPublishDate() {
        return publishDate;
    }

    public void setPublishDate(Date publishDate) {
        this.publishDate = publishDate;
    }
}
```

4.2 controller

```java
package com.itqf.controller;

import com.itqf.pojo.Book;
import com.itqf.service.BookService;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import org.springframework.web.bind.annotation.*;

import javax.annotation.Resource;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Created by liliting on 18/3/14.
 */
//@Controller
@RestController   //Controller+ResponseBody
@RequestMapping("/api1")
```

```
@Api(description = "图书管理api",value = "管理图书")
public class BookController {
    @Resource
    private BookService bookService;



    @ApiOperation(value = "查询所有图书")
    @GetMapping("/findAll")
    public List<Book> findAll()
    {
        return  bookService.findAll();
    }

    @ApiOperation(value = "新增图书")

    @PostMapping("/saveBook")
    public Book  save(@ApiParam(value = "图书对象") @RequestBody  Book book){//json
--》 对象
        bookService.saveBook(book);
        return book;
    }
    @ApiOperation(value = "修改图书",response = Book.class)
    @PutMapping("/updateBook")
    public Book  update(@ApiParam(value = "图书对象") Book book){
         bookService.update(book);
        return book;
    }

    @ApiOperation(value = "删除图书")
    @DeleteMapping("/deleteBook")
    public Map  delete(@ApiParam(required = true,value ="图书编号")  int id){
        Map map =  new HashMap();
      try{
          bookService.delete(id);
          map.put("result","success");
      }catch (Exception e){

          map.put("result","fail");


      }
        return map;
    }





}
```

4.3 添加swagger文档注释后效果


localhost:8088/book/swagger-ui.html#/

BookStore Platform API

BookStore Platform's REST API, all the applications could access the Object model data via JSON.

Created by java
See more at http://itqf.com
Contact the developer
The Apache License, Version 2.0

basic-error-controller : Basic Error Controller

book-controller : 图书管理api

DELETE /api1/deleteBook 删除图书

GET /api1/findAll 查询所有图书

Response Class (Status 200)
OK

Model Example Value

```
[
  {
    "author": "金庸",
    "id": 1,
    "name": "天龙八部",
    "price": 250,
    "publishDate": "2018-03-20"
  }
]
```

Response Content Type */*

有了中文的描述，是不是更加清晰了呢。