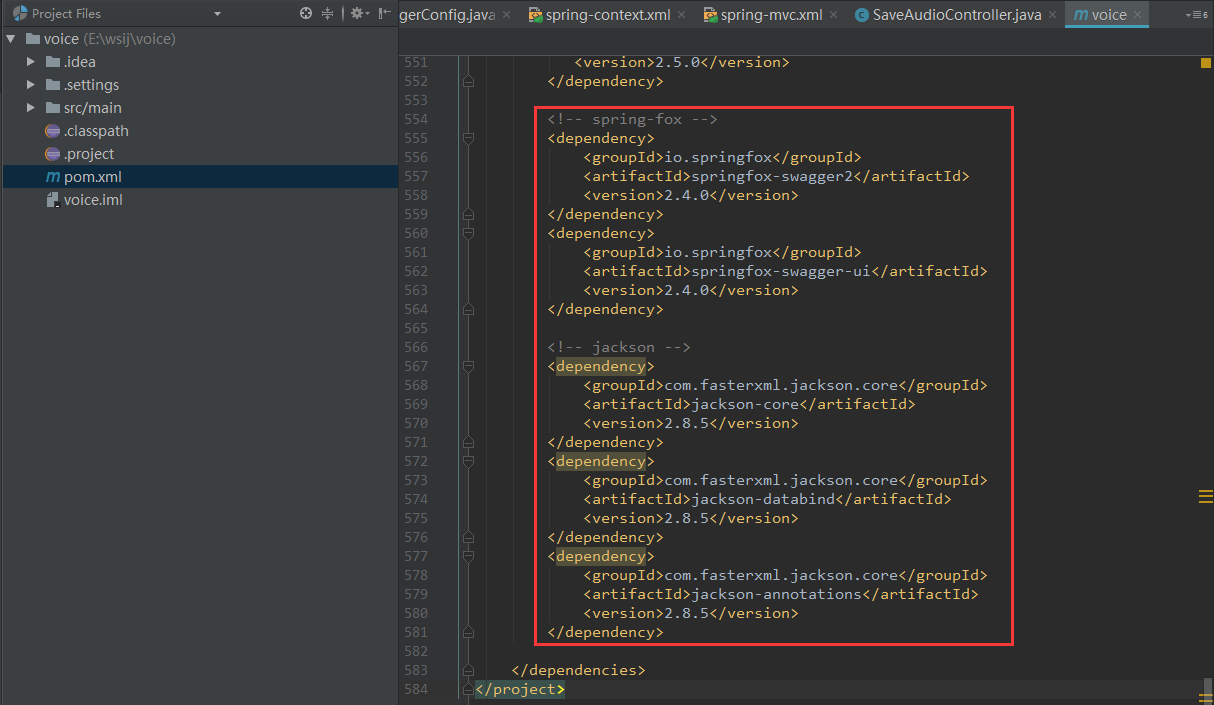
使用spring-fox配合swagger-ui形成动态api文档

## 在maven中引用插件：（spring-fox和fasterxml）

打开当前项目的maven配置文件（根路径下的pom.xml）

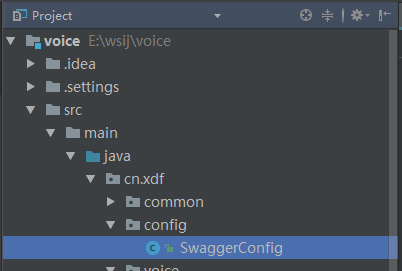


复制粘贴用：

<!-- spring-fox -->  
<dependency>  
 <groupId>io.springfox</groupId>  
 <artifactId>springfox-swagger2</artifactId>  
 <version>2.4.0</version>  
</dependency>  
<dependency>  
 <groupId>io.springfox</groupId>  
 <artifactId>springfox-swagger-ui</artifactId>  
 <version>2.4.0</version>  
</dependency>  
  
<!-- fasterxml -->  
<dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-core</artifactId>  
 <version>2.6.5</version>  
</dependency>  
<dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 <version>2.6.5</version>  
</dependency>  
<dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-annotations</artifactId>  
 <version>2.6.5</version>  
</dependency>  
<dependency>  
 <groupId>com.fasterxml</groupId>  
 <artifactId>classmate</artifactId>  
 <version>1.3.1</version>  
</dependency>

## 配置Swagger

在src/main/java下建立config文件夹，新建一个叫SwaggerConfig.java的文件

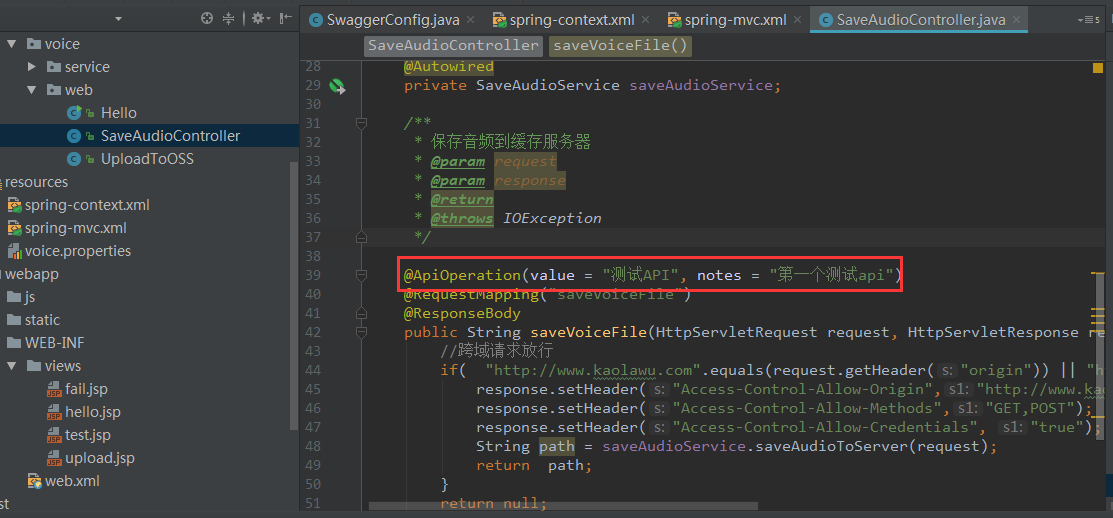


编辑文件为

package cn.xdf.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import springfox.documentation.builders.ApiInfoBuilder;  
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;  
  
@Configuration //必须存在  
@EnableSwagger2 //必须存在  
@EnableWebMvc //必须存在  
@ComponentScan(basePackages = {"cn.xdf.voice"})//必须存在 扫描的API Controller package name 也可以直接扫描class (basePackageClasses)  
public class SwaggerConfig {  
 @Bean  
 public Docket customDocket() {  
 return new Docket(DocumentationType.*SWAGGER\_2*)  
 .apiInfo(apiInfo());  
 }  
  
 private ApiInfo apiInfo() {  
 Contact contact = new Contact("小明", "http://www.cnblogs.com/getupmorning/", "zhaoming0018@126.com");  
 return new ApiInfoBuilder()  
 .title("前台API接口")  
 .description("前台API接口")  
 .contact(contact)  
 .version("1.1.0")  
 .build();  
 }  
}

发现无法自动解决依赖，请参照**附录一**

在需要暴露的地方加上类似代码，详细参看下文描述



## 暴露接口注解的详情

【以下内容摘抄自：http://blog.csdn.net/u012476983/article/details/54090423】

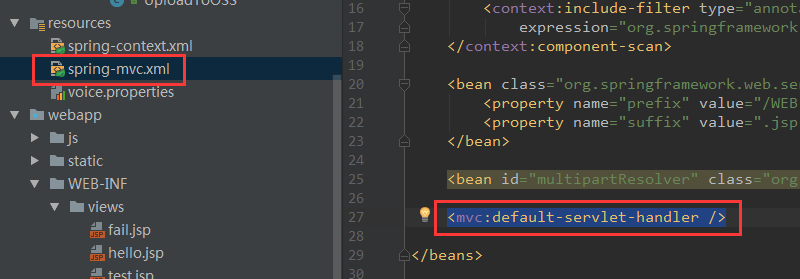
####springfox、swagger.annotations.\*注解部分参数介绍   
在上面只展示了如何使用，这里将对上面添加的swagger注解进行说明，笔记使用时参考了[swagger annotations Api 手册](http://docs.swagger.io/swagger-core/apidocs/com/wordnik/swagger/annotations/Api.html)，接下来进行部分常用注解使用说明介绍。   
- @ApiIgnore 忽略注解标注的类或者方法，不添加到API文档中

* @ApiOperation 展示每个API基本信息
  + value api名称
  + notes 备注说明
* @ApiImplicitParam 用于规定接收参数类型、名称、是否必须等信息
  + name 对应方法中接收参数名称
  + value 备注说明
  + required 是否必须 boolean
  + paramType 参数类型 body、path、query、header、form中的一种   
    - body 使用@RequestBody接收数据 POST有效
    - path 在url中配置{}的参数
    - query 普通查询参数 例如 ?query=q ,jquery ajax中data设置的值也可以，例如 {query:”q”},springMVC中不需要添加注解接收
    - header 使用@RequestHeader接收数据
    - form 笔者未使用，请查看官方API文档
  + dataType 数据类型，如果类型名称相同，请指定全路径，例如 dataType = “java.util.Date”，springfox会自动根据类型生成模型
* @ApiImplicitParams 包含多个@ApiImplicitParam
* @ApiModelProperty 对模型中属性添加说明，例如 上面的PageInfoBeen、BlogArticleBeen这两个类中使用，只能使用在类中。
  + value 参数名称
  + required 是否必须 boolean
  + hidden 是否隐藏 boolean   
    其他信息和上面同名属性作用相同，hidden属性对于集合不能隐藏，目前不知道原因
* @ApiParam 对单独某个参数进行说明，使用在类中或者controller方法中都可以。注解中的属性和上面列出的同名属性作用相同

以上为主要常用的注解介绍，请结合springfox使用查看

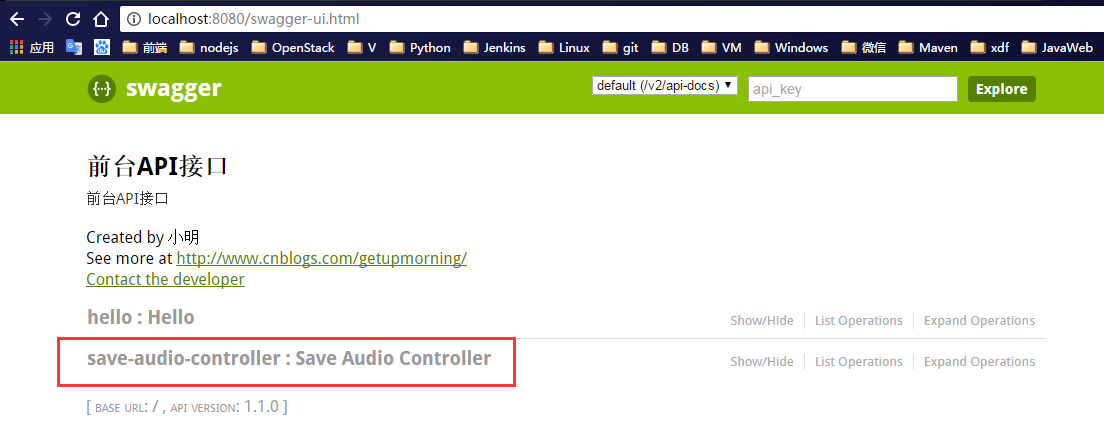
【以上内容摘抄自：http://blog.csdn.net/u012476983/article/details/54090423】

## 配置spring-mvc.xml，使访问静态文件绕过过滤器

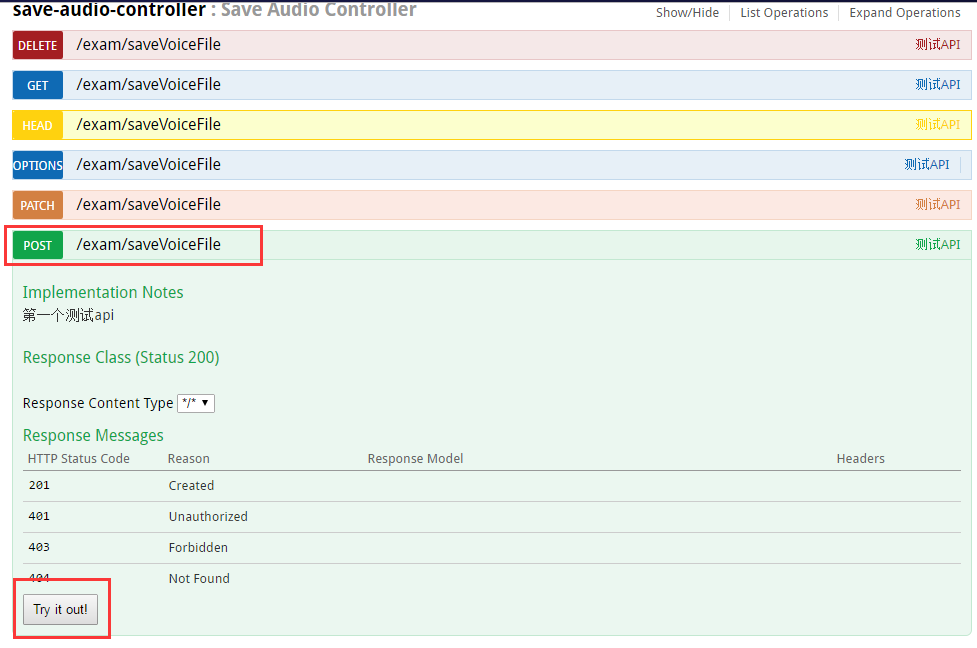


## 启动项目，查看结果

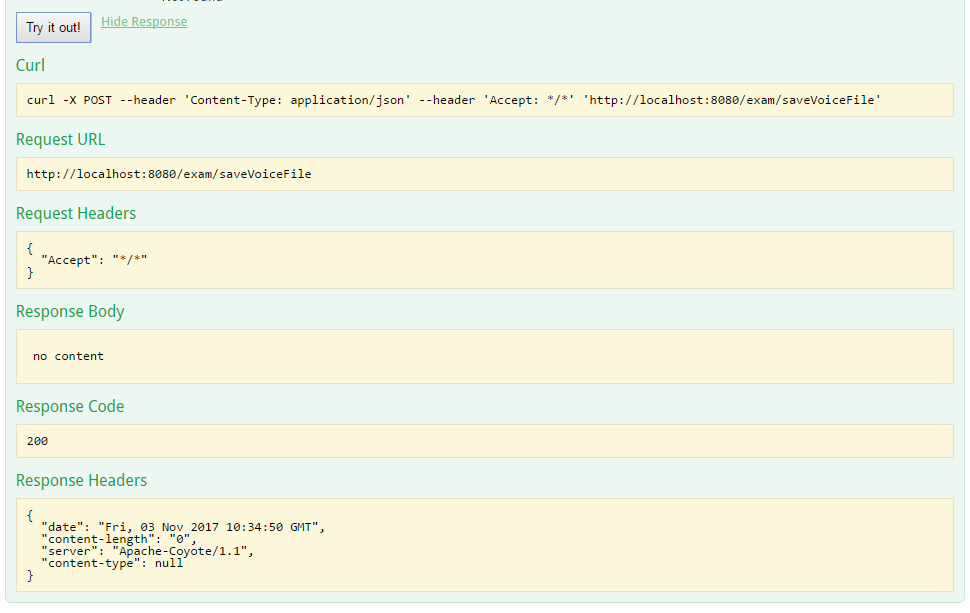
访问地址：<http://localhost:8080/swagger-ui.html>



依次进入层级关系，可以看到之前配置的描述性文字



点击最下方的Try it out!

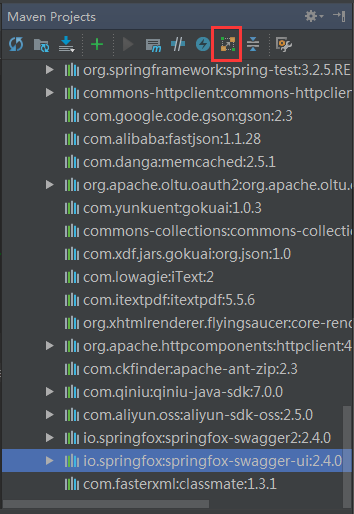


可以看到测试接口成功

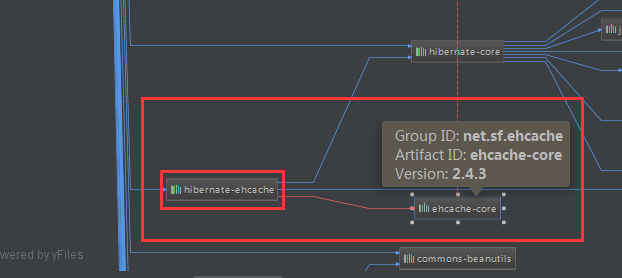
这里可以发现，同名方法有许多个，是因为未在注解接口的时候详细进行配置，当前这步只是简单测试一下swagger插件是否运行正常。

## 如果启动失败报错，是某个jar包里的某个方法未找到，可以进行如下操作解决：

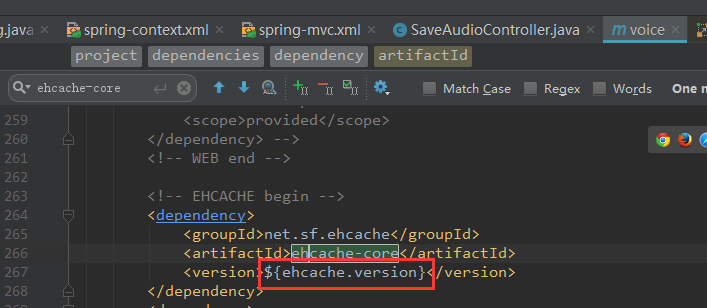
在Maven Projects视图窗口中点击Show Dependencies按钮



找到冲突部分



双击左侧包，打开这个Maven Project的pom.xml配置文件



查看当前这个“包”的版本号



发现与上冲突，于是更新“根路径下的pom.xml”对应位置的版本号，然后重新启动服务，重试。

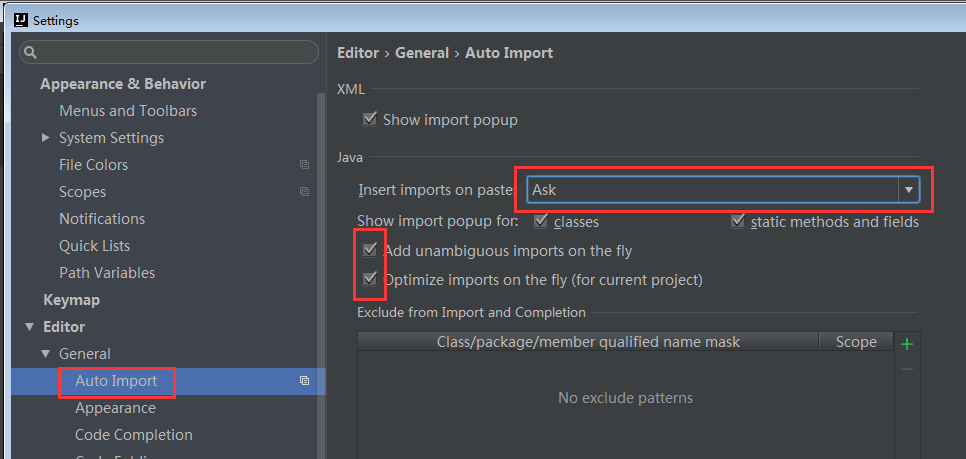


再次发现在此如上述处理。直到不报jar包找不到对应方法之类的错误。

## 附录一：IntelliJ的部分操作补充

### 自动解决依赖的设置

修改setting如下



注意：java文件只有放在java路径下才会自动解决依赖，如果有依赖重名的问题，需要使用<Alt> + <Enter>手动处理解决。

### 配置tomcat运行环境

参看此网页：http://www.cnblogs.com/shijiaoyun/p/5882281.html

## 附录二：参考文档

Swagger插件相关：

<http://blog.csdn.net/u012476983/article/details/54090423>

<http://blog.csdn.net/guanfl/article/details/54574443>

<http://blog.csdn.net/u014231523/article/details/54411026>

[推荐]<http://www.cnblogs.com/getupmorning/p/7267076.html>

Maven依赖冲突：

<http://blog.csdn.net/fengzheku/article/details/50688044>

IntelliJ配置tomcat运行环境：

<http://www.cnblogs.com/shijiaoyun/p/5882281.html>