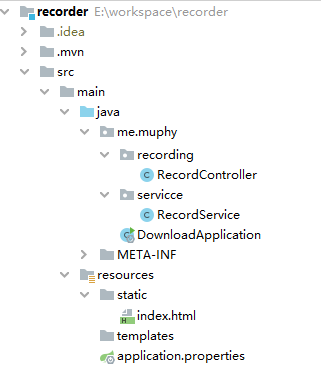
# 基于Java的录音机（Rest API的页面操作）

## 新建Spring Boot项目



## pop.xml

<?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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>2.2.5.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>me.muphy</groupId>

<artifactId>recorder</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>recorder</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

## application.properties

server.port=8080

#录音

record.time.default=3600

record.path=E:/workspace/share/

## index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>莫非录音机</title>

</head>

<body>

<div>

<form action="/record/start" method="get">

<div>请输入需要录音的时间(s):<input type="number" name="t" id="t" max="7200" min="0"><button type="submit">开始录音</button></div>

</form>

</div>

</body>

</html>

## RecordController.java

package me.muphy.recording;

import me.muphy.servicce.RecordService;

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

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

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

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

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

import javax.servlet.http.HttpServletRequest;

import javax.sound.sampled.\*;

import java.io.\*;

@RestController

@RequestMapping("/record")

public class RecordController {

@Autowired

private RecordService recordService;

@Value("${download.url:http://(ip):(port)}")

private String downloadUrl;

@GetMapping("/start")

public String startRecord(int t, HttpServletRequest request) {

String msg = recordService.start(t);

return "<span>" + msg + "</span><br><span><a href=\"/record/stop\" >点击停止录音</a></span>";

}

@GetMapping("/stop")

public String stopRecord(HttpServletRequest request) {

String msg = recordService.stop();

return "<span>" + msg + "</span>";

}

}

## RecordService.java

package me.muphy.servicce;

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

import org.springframework.stereotype.Service;

import javax.sound.sampled.\*;

import java.io.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Timer;

import java.util.TimerTask;

@Service

public class RecordService {

@Value("${download.path:E:/workspace/download/}")

private String downloadPath;

@Value("${record.time.default:3600}")

private int defaultRecordTime;

//定义停止录音的标志，来控制录音线程的运行

private static volatile boolean stopFlag = true;

public String start() {

return start(defaultRecordTime);

}

public String start(int time) {

synchronized (RecordService.class){

if(!stopFlag){

return "已有录音程序正在运行，启动录音失败!";

}

}

new Thread(new Record()).start();

time = ((time > 7200 || time == 0) ? defaultRecordTime : time);

Timer timer = new Timer();

timer.schedule(new TimerTask() {

@Override

public void run() {

stop();

}

}, 1000 \* time);

return "启动录音成功,开始录音," + time + "秒之后自动停止！";

}

//停止录音

public String stop() {

if(stopFlag){

return "没有运行的录音程序!";

}

stopFlag = true;

return "停止成功!";

}

//文件拷贝方法

public void copyFile(String srcPath, String destPath) {

File srcFile = new File(srcPath);

//如果目的文件夹没有则创建目的文件夹

(new File(destPath)).mkdirs();

//在目的文件夹下创建要复制的文件

File destFile = new File(destPath + "/" + srcFile.getName());

if (srcFile.isFile() && srcFile.exists()) {

InputStream in = null;

OutputStream out = null;

try {

in = new FileInputStream(srcFile);

out = new FileOutputStream(destFile);

//设置缓冲数组

byte[] buff = new byte[1024 \* 5];

int len = 0;

while ((len = in.read(buff)) != -1) {

out.write(buff, 0, len);

}

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

if (out != null) {

out.close();

}

if (in != null) {

in.close();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

//复制过后删除源文件夹中的的文件

if (srcFile.exists()) {

srcFile.delete();

}

}

//录音类，因为要用到MyRecord类中的变量，所以将其做成内部类

private class Record implements Runnable {

//定义存放录音的字节数组,作为缓冲区

byte bts[] = new byte[10000];

//定义录音格式

private AudioFormat af = null;

//定义目标数据行,可以从中读取音频数据,该 TargetDataLine 接口提供从目标数据行的缓冲区读取所捕获数据的方法。

private TargetDataLine td = null;

//定义源数据行,源数据行是可以写入数据的数据行。它充当其混频器的源。应用程序将音频字节写入源数据行，这样可处理字节缓冲并将它们传递给混频器。

private SourceDataLine sd = null;

//定义字节数组输入输出流

private ByteArrayInputStream bais = null;

private ByteArrayOutputStream baos = null;

//定义音频输入流

private AudioInputStream ais = null;

//将字节数组包装到流里，最终存入到baos中

//重写run函数

@Override

public void run() {

//af为AudioFormat也就是音频格式

af = getAudioFormat();

DataLine.Info info = new DataLine.Info(TargetDataLine.class, af);

baos = new ByteArrayOutputStream();

try {

stopFlag = false;

td = (TargetDataLine) (AudioSystem.getLine(info));

//打开具有指定格式的行，这样可使行获得所有所需的系统资源并变得可操作。

td.open(af);

//允许某一数据行执行数据 I/O

td.start();

while (!stopFlag) {

//当停止录音没按下时，该线程一直执行

//从数据行的输入缓冲区读取音频数据。

//要读取bts.length长度的字节,cnt 是实际读取的字节数

int cnt = td.read(bts, 0, bts.length);

if (cnt > 0) {

baos.write(bts, 0, cnt);

}

//开始从音频流中读取字节数

byte copyBts[] = bts;

bais = new ByteArrayInputStream(copyBts);

ais = new AudioInputStream(bais, af, copyBts.length / af.getFrameSize());

try {

DataLine.Info dataLineInfo = new DataLine.Info(SourceDataLine.class, af);

sd = (SourceDataLine) AudioSystem.getLine(dataLineInfo);

sd.open(af);

sd.start();

} catch (Exception e) {

e.printStackTrace();

}

}

save();

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

//关闭打开的字节数组流

if (baos != null) {

baos.close();

}

if (bais != null) {

bais.close();

}

if (ais != null) {

ais.close();

}

} catch (Exception e) {

e.printStackTrace();

} finally {

td.close();

}

}

}

private void save() throws IOException {

byte audioData[] = baos.toByteArray();

bais = new ByteArrayInputStream(audioData);

ais = new AudioInputStream(bais, af, audioData.length / af.getFrameSize());

//定义最终保存的文件名

File file = null;

//以当前的时间命名录音的名字

String filePath = downloadPath + "/record";

File path = new File(filePath);

if (!path.exists()) {//如果文件不存在，则创建该目录

path.mkdirs();

}

String time = new SimpleDateFormat("yyyMMddHHmmss").format(new Date());

file = new File(filePath + "/" + time + ".wav");

AudioSystem.write(ais, AudioFileFormat.Type.WAVE, file);

}

//设置AudioFormat的参数

private AudioFormat getAudioFormat() {

//下面注释部分是另外一种音频格式，两者都可以

AudioFormat.Encoding encoding = AudioFormat.Encoding.PCM\_SIGNED;

// 采样率是每秒播放和录制的样本数 8000,11025,16000,22050,44100

float sampleRate = 8000f;

// 每个具有此格式的声音样本中的位数 8,16

int sampleSizeInBits = 16;

String signedString = "signed";

boolean bigEndian = true;

// 单声道为1，立体声为2

int channels = 1;

return new AudioFormat(encoding, sampleRate, sampleSizeInBits, channels,

(sampleSizeInBits / 8) \* channels, sampleRate, bigEndian);

}

}

}

## RecordApplication.java

package me.muphy;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RecordApplication {

public static void main(String[] args) {

SpringApplication.run(RecordApplication.class, args);

}

}

## 启动项目

