# SignSystemApplication.java

package com.isl;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.scheduling.annotation.EnableScheduling;  
  
@SpringBootApplication  
@EnableScheduling  
public class SignSystemApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(SignSystemApplication.class, args);  
 }  
  
}

# CheckAdvice.java

package com.isl.advice;  
  
  
import java.io.UnsupportedEncodingException;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.ControllerAdvice;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.bind.support.WebExchangeBindException;  
  
import com.isl.advice.exceptions.CheckException;  
  
/\*\*  
 \* 异常处理切面  
 \* @author Jason  
 \*  
 \*/  
@ControllerAdvice  
public class CheckAdvice {  
  
 @ExceptionHandler(CheckException.class)  
 public ResponseEntity<String> handleBindException(CheckException e){  
 return new ResponseEntity<String>("错误的字段:"+e.getLocation()+"错误的原因:"+e.getReason(),HttpStatus.BAD\_REQUEST);  
 }  
   
   
 @ExceptionHandler(WebExchangeBindException.class)  
 public ResponseEntity<String> handleBindException(WebExchangeBindException e){  
 return new ResponseEntity<String>(e.getFieldErrors().stream().map(ex->ex.getField()+":"+ex.getDefaultMessage())  
 .reduce("",(s1,s2)->s1+"\n"+s2),HttpStatus.BAD\_REQUEST);  
 }  
   
 /\*\*  
 \* 用户身份范围异常  
 \* @param e  
 \* @return  
 \*/  
 @ExceptionHandler(IllegalArgumentException.class)  
 public ResponseEntity<String> handleBindException(IllegalArgumentException e){  
 return new ResponseEntity<String>(e.getMessage(),HttpStatus.BAD\_REQUEST);  
 }  
  
  
 /\*\*  
 \* String转byte错误异常  
 \* @param e  
 \* @return  
 \*/  
 @ExceptionHandler(UnsupportedEncodingException.class)  
 public ResponseEntity<String> handleBindException(UnsupportedEncodingException e){  
 return new ResponseEntity<String>(e.getMessage(),HttpStatus.BAD\_REQUEST);  
 }  
   
}

# CheckException.java

package com.isl.advice.exceptions;  
  
import lombok.Data;  
  
@Data  
public class CheckException extends RuntimeException {  
  
 private static final long serialVersionUID = 1L;  
  
 /\*\*  
 \* 出错的字段  
 \*/  
 private String location;  
 /\*\*  
 \* 出错原因  
 \*/  
 private String reason;  
   
   
 public CheckException() {  
 super();  
 }  
  
 public CheckException(String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace) {  
 super(message, cause, enableSuppression, writableStackTrace);  
 }  
  
 public CheckException(String message, Throwable cause) {  
 super(message, cause);  
 }  
  
 public CheckException(String message) {  
 super(message);  
 }  
  
 public CheckException(Throwable cause) {  
 super(cause);  
 }  
  
 public CheckException(String location, String reason) {  
 super();  
 this.location = location;  
 this.reason = reason;  
 }  
   
}

# QuartzConfig.java

package com.isl.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.scheduling.quartz.JobDetailFactoryBean;  
import org.springframework.scheduling.quartz.SchedulerFactoryBean;  
import org.springframework.scheduling.quartz.SimpleTriggerFactoryBean;  
  
import com.isl.quartz.MyAdaptableJobFactory;  
import com.isl.quartz.QuartzOne;  
  
@Configuration  
public class QuartzConfig {  
 /\*\*\*  
 \* 1创建Job对象  
 \*/  
 @Bean  
 public JobDetailFactoryBean jobDetailFactoryBean()  
 {  
 JobDetailFactoryBean factoryBean =new JobDetailFactoryBean();  
 factoryBean.setJobClass(QuartzOne.class);  
 return factoryBean;  
 }  
 /\*\*\*  
 \* 2创建Trigger对象  
 \*/  
 @Bean  
 public SimpleTriggerFactoryBean simpleTriggerFactoryBean(JobDetailFactoryBean jobDetailFactoryBean) {  
 SimpleTriggerFactoryBean factoryBean = new SimpleTriggerFactoryBean();  
 factoryBean.setJobDetail(jobDetailFactoryBean.getObject());  
 //执行间隔 毫秒  
 factoryBean.setRepeatInterval(2000);  
 //重复次数  
 //factoryBean.setRepeatCount(0);  
 return factoryBean;  
 }  
   
   
   
 /\*\*\*  
 \* 3创建Scheduler对象  
 \*/  
 @Bean  
 public SchedulerFactoryBean schedulerFactoryBean(SimpleTriggerFactoryBean simpleTriggerFactoryBean,MyAdaptableJobFactory myAdaptableJobFactory)  
 {  
 SchedulerFactoryBean schedulerFactoryBean= new SchedulerFactoryBean();  
 schedulerFactoryBean.setTriggers(simpleTriggerFactoryBean.getObject());  
 schedulerFactoryBean.setJobFactory(myAdaptableJobFactory);  
 return schedulerFactoryBean;  
 }  
  
}

# SwaggerConfig.java

package com.isl.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
  
import io.swagger.annotations.ApiOperation;  
import springfox.documentation.builders.RequestHandlerSelectors;  
import springfox.documentation.spi.DocumentationType;  
import springfox.documentation.spring.web.plugins.Docket;  
import springfox.documentation.swagger2.annotations.EnableSwagger2;  
  
@Configuration  
@EnableSwagger2  
public class SwaggerConfig{  
 @Bean  
 public Docket swaggerSpringMvcPlugin() {  
 return new Docket(DocumentationType.SWAGGER\_2)  
 .select()  
 .apis(RequestHandlerSelectors.withMethodAnnotation(ApiOperation.class))  
 .build();  
 }  
}

# WebConfig.java

package com.isl.config;  
  
import java.util.Date;  
import java.util.Locale;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.format.Formatter;  
import org.springframework.format.FormatterRegistry;  
import org.springframework.web.servlet.config.annotation.CorsRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
  
@Configuration  
public class WebConfig implements WebMvcConfigurer {  
 @Override  
 public void addFormatters(FormatterRegistry registry) {  
 registry.addFormatter(new Formatter<Date>() {  
 @Override  
 public Date parse(String date, Locale locale) {  
 return new Date(Long.parseLong(date));  
 }  
  
 @Override  
 public String print(Date date, Locale locale) {  
 return Long.valueOf(date.getTime()).toString();  
 }  
 });  
 }  
   
// @Override   
 //允许所有跨域访问  
// public void addCorsMappings(CorsRegistry registry) {   
// registry.addMapping(" /\*\*");  
// }  
}

# AcademyController.java

package com.isl.controller;  
  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
import org.apache.commons.lang3.StringUtils;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
import com.isl.entity.Academy;  
import com.isl.entity.request.AcademyReq;  
import com.isl.entity.response.AcademyResp;  
import com.isl.excel.ExcelFileReader;  
import com.isl.service.AcademyRepository;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "学院信息相关接口")  
@RestController  
@RequestMapping("/academy")  
public class AcademyController {  
 private AcademyRepository academyRepository;  
  
 /\*\*  
 \* 构造函数注入  
 \*   
 \* @param academyRepository  
 \*/  
 public AcademyController(AcademyRepository academyRepository) {  
 super();  
 this.academyRepository = academyRepository;  
 }  
  
 @ApiOperation(value = "获取全部学院信息", notes = "获取全部学院信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = AcademyResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/all")  
 public List<AcademyResp> getAll() {  
 return academyRepository.findAll().stream().map(entity -> new AcademyResp(entity)).collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增学院信息", notes = "上传必要的学院信息信息来创建一个新的学院信息")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = AcademyResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/add")  
 public AcademyResp add(  
 @ApiParam(value = "需要更新的课时信息,以json格式放入Request Body中", required = true) @Valid @RequestBody AcademyReq academyReq,  
 HttpSession session) {  
  
 Academy academy = new Academy(academyReq);  
 // 学院信息自定义完整性进行校验  
 AcademyCheck(academy);  
 String userId = (String) session.getAttribute("userId");  
 Date nowTime = new Date();  
 academy.setCreateTime(nowTime);  
 academy.setUpdateTime(nowTime);  
 academy.setCreateBy(userId);  
 academy.setUpdateBy(userId);  
 academyRepository.save(academy);  
 return new AcademyResp(academy);  
  
 }  
  
 @ApiOperation(value = "删除学院信息", notes = "根据学院信息的academy\_id来删除一个学院信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "academy\_id", value = "被操作的目标主键,直接放入地址中,替换{academy\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{academy\_id}")  
 public ResponseEntity<Void> delete(@PathVariable("academy\_id") String academy\_id, HttpSession session) {  
 return academyRepository.findById(academy\_id).map(entity -> {  
 academyRepository.delete(entity);  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 }).orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "更新学院信息信息", notes = "通过academy\_id定位学院信息并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "academy\_id", value = "被操作的目标主键,直接放入地址中,替换{academy\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = AcademyResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{academy\_id}")  
 public ResponseEntity<AcademyResp> update(@PathVariable("academy\_id") String academy\_id,  
 @ApiParam(value = "需要更新的课时信息,以json格式放入Request Body中", required = true) @RequestBody AcademyReq academyReq,  
 HttpSession session) {  
  
 Academy academy = new Academy(academyReq);  
  
 return academyRepository.findById(academy\_id).map(entity -> {  
 if (StringUtils.isNotBlank(academy.getAcademyName())) {  
 entity.setAcademyName(academy.getAcademyName());  
 }  
 String userId = (String) session.getAttribute("userId");  
 Date nowTime = new Date();  
 entity.setUpdateTime(nowTime);  
 entity.setUpdateBy(userId);  
 AcademyCheck(entity);  
  
 academyRepository.save(entity);  
 return entity;  
 }).map(entity -> new ResponseEntity<AcademyResp>(new AcademyResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "根据主键查找学院信息", notes = "根据学院信息academy\_id查找学院信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "academy\_id", value = "被操作的目标主键,直接放入地址中,替换{academy\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = AcademyResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{academy\_id}")  
 public ResponseEntity<AcademyResp> findByID(@PathVariable("academy\_id") String academy\_id) {  
 return academyRepository.findById(academy\_id)  
 .map(entity -> new ResponseEntity<AcademyResp>(new AcademyResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "大批量增加学院信息", notes = "上传学院信息表格,以List一次性添加数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("addAcademyByExcel")  
 public ArrayList<AcademyResp> addAcademyByFile(@RequestParam("file") MultipartFile file, HttpSession session) {  
 if (file == null)  
 return null;  
 String ofname = file.getOriginalFilename();  
 if (ofname == null || ("").equals(ofname) && file.getSize() == 0)  
 return null;  
 ArrayList<AcademyResp> academyresps = new ArrayList<AcademyResp>();  
 List<AcademyReq> reqlist = ExcelFileReader.readfile(file, new AcademyReq());  
 for (AcademyReq academyReq : reqlist) {  
 System.out.println(academyReq.toString());  
 Academy academy = new Academy(academyReq);  
 // 学院信息自定义完整性进行校验  
 AcademyCheck(academy);  
 String userId = (String) session.getAttribute("userId");  
 Date nowTime = new Date();  
 academy.setCreateTime(nowTime);  
 academy.setUpdateTime(nowTime);  
 academy.setCreateBy(userId);  
 academy.setUpdateBy(userId);  
 academyresps.add(new AcademyResp(academy));  
 academyRepository.save(academy);  
 }  
 return academyresps.isEmpty() ? null : academyresps;  
 }  
  
 // 学院信息自定义完整性进行校验，校验有问题就抛异常  
 private void AcademyCheck(@Valid Academy entity) {  
  
 }  
  
}

# ClassController.java

package com.isl.controller;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
import com.isl.entity.Class;  
import com.isl.entity.request.ClassReq;  
import com.isl.entity.response.ClassResp;  
import com.isl.excel.ExcelFileReader;  
import com.isl.service.ClassRepository;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "班级信息相关接口")  
@RestController  
@RequestMapping("/class")  
public class ClassController {  
  
 private ClassRepository classRepository;  
  
 /\*\*  
 \* 构造函数注入  
 \*   
 \* @param majorRepository  
 \*/  
 public ClassController(ClassRepository classRepository) {  
 super();  
 this.classRepository = classRepository;  
 }  
  
 @ApiOperation(value = "获取全部班级信息", notes = "获取全部班级信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = ClassResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/allClass")  
 public List<ClassResp> getAllClass() {  
 return classRepository.findAll().stream().map(entity -> new ClassResp(entity)).collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增班级信息", notes = "上传必要的班级信息来创建一条新的班级记录")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = ClassResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/addClass")  
 public ClassResp addClass(  
 @ApiParam(value = "需要创建的班级记录，以json格式放入Request body中", required = true) @Valid @RequestBody ClassReq classReq,  
 HttpSession session) {  
 Class class1 = new Class(classReq);  
 Check(class1);  
 String updateID = (String) session.getAttribute("updateID");  
 class1.setCreateBy(updateID);  
 class1.setUpdateBy(updateID);  
 class1.setCreateTime(System.currentTimeMillis() / 1000);  
 class1.setUpdateTime(System.currentTimeMillis() / 1000);  
 classRepository.save(class1);  
 return new ClassResp(class1);  
 }  
  
 @ApiOperation(value = "删除班级记录", notes = "根据班级记录的classid来删除一个选课记录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "classid", value = "被操作的目标主键,直接放入地址中,替换{classid}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{classid}")  
 public ResponseEntity<Void> deleteClass(@PathVariable("classid") String classid, HttpSession session) {  
 return classRepository.findById(classid).map(entity -> {  
 classRepository.delete(entity);  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 }).orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "更新班级记录", notes = "通过classid定位班级记录并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "classid", value = "被操作的目标主键,直接放入地址中,替换{classid}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = ClassResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{classid}")  
 public ResponseEntity<ClassResp> updateClass(@PathVariable("classid") String classid,  
 @ApiParam(value = "需要更新的选课记录信息,以json格式放入Request Body中", required = true) @RequestBody ClassReq classReq,  
 HttpSession session) {  
 Class newclass = new Class(classReq);  
 Class class1 = classRepository.findByClassID(classid);  
 if (newclass.getClassName() != null)  
 class1.setClassName(class1.getClassName());  
 if (newclass.getMajorID() != null)  
 class1.setMajorID(newclass.getMajorID());  
 String updateID = (String) session.getAttribute("updateID");  
 class1.setUpdateBy(updateID);  
 class1.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (classRepository.save(class1) != null)  
 return new ResponseEntity<ClassResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "根据主键查找班级记录", notes = "根据班级记录主键majorid查找记录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "classid", value = "被操作的目标主键,直接放入地址中,替换{classid}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = ClassResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{classid}")  
 public ResponseEntity<ClassResp> findClassByID(@PathVariable("classid") String classid) {  
 return classRepository.findById(classid)  
 .map(entity -> new ResponseEntity<ClassResp>(new ClassResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "大批量增加班级记录信息", notes = "上传班级信息表格,以List一次性添加数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("addClassByExcel")  
 public List<ClassResp> addClassByFile(@RequestParam("file") MultipartFile file, HttpSession session) {  
 if (file == null)  
 return null;  
 String ofname = file.getOriginalFilename();  
 if (ofname == null || ("").equals(ofname) && file.getSize() == 0)  
 return null;  
 ArrayList<ClassResp> classresps = new ArrayList<ClassResp>();  
 List<ClassReq> reqList = ExcelFileReader.readfile(file, new ClassReq());  
 for (ClassReq req : reqList) {  
 System.out.println(req.toString());  
 Class class1 = new Class(req);  
 System.out.println(class1.toString());  
 String updateID = (String) session.getAttribute("updateID");  
 class1.setCreateTime(System.currentTimeMillis() / 1000);  
 class1.setUpdateTime(System.currentTimeMillis() / 1000);  
 class1.setCreateBy(updateID);  
 class1.setUpdateBy(updateID);  
 System.out.println(class1.toString());  
 classresps.add(new ClassResp(class1));  
 classRepository.save(class1);  
 }  
 return classresps.isEmpty() ? null : classresps;  
 }  
  
 public void Check(@Valid Class entity) {  
  
 }  
  
}

# CourseController.java

package com.isl.controller;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
import com.isl.entity.Course;  
import com.isl.entity.request.CourseReq;  
import com.isl.entity.response.CourseResp;  
import com.isl.excel.ExcelFileReader;  
import com.isl.service.CourseRepository;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "课程信息相关接口")  
@RestController  
@RequestMapping("/course")  
public class CourseController {  
 private CourseRepository courseRepository;  
  
 /\*\*  
 \* 构造函数注入  
 \*   
 \* @param courseRepository  
 \*/  
 public CourseController(CourseRepository courseRepository) {  
 super();  
 this.courseRepository = courseRepository;  
 }  
  
 @ApiOperation(value = "获取全部课程信息", notes = "获取全部课程信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = CourseResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/allCourse")  
 public List<CourseResp> getAllCourse() {  
 return courseRepository.findAll()// 获取所有课程信息  
 .stream().map(entity -> new CourseResp(entity)).collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增课程信息", notes = "上传必要的课程信息来创建一门新的课程")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = CourseResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/addCourse")  
 public CourseResp addCourse(  
 @ApiParam(value = "需要创建的课程信息，以json格式放入Request body中", required = true) @Valid @RequestBody CourseReq courseReq,  
 HttpSession session) {  
 Course course = new Course(courseReq);  
 CourseCheck(course);  
 String updateID = (String) session.getAttribute("updateID");  
 course.setCreateTime(System.currentTimeMillis() / 1000);  
 course.setUpdateTime(System.currentTimeMillis() / 1000);  
 course.setCreateBy(updateID);  
 course.setUpdateBy(updateID);  
 courseRepository.save(course);  
 return new CourseResp(course);  
  
 }  
  
 @ApiOperation(value = "删除课程信息", notes = "根据用户信息的course\_id来删除一个课程信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "course\_id", value = "被操作的目标主键,直接放入地址中,替换{course\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{course\_id}")  
 public ResponseEntity<Void> deleteCourse(@PathVariable("course\_id") String course\_id, HttpSession session) {  
 return courseRepository.findById(course\_id).map(entity -> {  
 courseRepository.delete(entity);  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 }).orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "更新课程信息", notes = "通过course\_id定位课程信息并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "course\_id", value = "被操作的目标主键,直接放入地址中,替换{course\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = CourseResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{course\_id}")  
 public ResponseEntity<CourseResp> updateCourse(@PathVariable("course\_id") String course\_id,  
 @ApiParam(value = "需要更新的课程信息,以json格式放入Request Body中", required = true) @RequestBody CourseReq courseReq,  
 HttpSession session) {  
 Course newCourse = new Course(courseReq);  
 Course course = courseRepository.findByCourseID(course\_id);  
 if (newCourse.getDescribeID() != null)  
 course.setDescribeID(newCourse.getDescribeID());  
 if (newCourse.getTeacherID() != null)  
 course.setTeacherID(newCourse.getTeacherID());  
 if (newCourse.getAdministratorID() != null)  
 course.setAdministratorID(newCourse.getAdministratorID());  
 if (newCourse.getMajorName() != null)  
 course.setMajorName(newCourse.getMajorName());  
 if (newCourse.getCourseName() != null)  
 course.setCourseName(newCourse.getCourseName());  
 if (newCourse.getBeginTime() != null)  
 course.setBeginTime(newCourse.getBeginTime());  
 if (String.valueOf(newCourse.getHours()) != null)  
 course.setHours(newCourse.getHours());  
 if (newCourse.getCourseLocation() != null)  
 course.setCourseLocation(newCourse.getCourseLocation());  
 String updateID = (String) session.getAttribute("updateID");  
 course.setUpdateBy(updateID);  
 course.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (courseRepository.save(course) != null)  
 return new ResponseEntity<CourseResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "根据主键查找课程信息", notes = "根据课程信息course\_id查找课程信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "course\_id", value = "被操作的目标主键,直接放入地址中,替换{course\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = CourseResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{course\_id}")  
 public ResponseEntity<CourseResp> findByID(@PathVariable("course\_id") String course\_id) {  
 return courseRepository.findById(course\_id)  
 .map(entity -> new ResponseEntity<CourseResp>(new CourseResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "大批量增加课程信息", notes = "上传课程信息表格,以List一次性添加数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("addCourseByExcel")  
 public List<CourseResp> addCourseByFile(@RequestParam("file") MultipartFile file, HttpSession session) {  
 if (file == null)  
 return null;  
 String ofname = file.getOriginalFilename();  
 if (ofname == null || ("").equals(ofname) && file.getSize() == 0)  
 return null;  
 ArrayList<CourseResp> courseresps = new ArrayList<CourseResp>();  
 List<CourseReq> reqList = ExcelFileReader.readfile(file, new CourseReq());  
 for (CourseReq courseReq : reqList) {  
 System.out.println(courseReq.toString());  
 Course course = new Course(courseReq);  
 System.out.println(course.toString());  
 String updateID = (String) session.getAttribute("updateID");  
 course.setCreateTime(System.currentTimeMillis() / 1000);  
 course.setUpdateTime(System.currentTimeMillis() / 1000);  
 course.setCreateBy(updateID);  
 course.setUpdateBy(updateID);  
 System.out.println(course.toString());  
 courseresps.add(new CourseResp(course));  
 courseRepository.save(course);  
 }  
 return courseresps.isEmpty() ? null : courseresps;  
 }  
  
 @ApiOperation(value = "获取该教师全部课程信息", notes = "获取该教师全部课程信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/allCByTid")  
 public List<CourseResp> getAllCourseByTeaid(  
 @ApiParam(value = "需要查询的教师id，放入Request body中", required = true) @RequestBody String tea\_id) {  
 return courseRepository.findCoursesByTeaId(tea\_id)// 获取该教师所有课程信息  
 .stream().map(entity -> new CourseResp(entity)).collect(Collectors.toList());  
 }  
  
 // 课程信息自定义完整性校验，有问题就抛异常  
 public void CourseCheck(@Valid Course entity) {  
  
 }  
}

# Course\_describeController.java

package com.isl.controller;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
import com.isl.entity.Course\_describe;  
import com.isl.entity.request.Course\_describeReq;  
import com.isl.entity.response.Course\_describeResp;  
import com.isl.excel.ExcelFileReader;  
import com.isl.service.Course\_describeRepository;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "课程信息描述相关接口")  
@RestController  
@RequestMapping("/course\_describe")  
public class Course\_describeController {  
 private Course\_describeRepository course\_describeRepository;  
  
 public Course\_describeController(Course\_describeRepository course\_describeRepository) {  
 super();  
 this.course\_describeRepository = course\_describeRepository;  
 }  
  
 /\*\*  
 \* 构造函数注入  
 \*   
 \* @param course\_describeRepository  
 \* @return  
 \*/  
 @ApiOperation(value = "获取全部课程描述信息", notes = "获取全部课程描述信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = Course\_describeResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/allcd")  
 public List<Course\_describeResp> getAllCd() {  
 return course\_describeRepository.findAll()// 获取所有课程信息  
 .stream().map(entity -> new Course\_describeResp(entity)).collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增课程描述信息", notes = "上传必要的课程描述信息来创建一门新的课程描述信息")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = Course\_describeResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/addCd")  
 public Course\_describeResp addCd(  
 @ApiParam(value = "需要创建的课程描述信息，以json格式放入Request body中", required = true) @Valid @RequestBody Course\_describeReq course\_describeReq,  
 HttpSession session) {  
 Course\_describe cd = new Course\_describe(course\_describeReq);  
 System.out.println(cd.getCourseBeginTime());  
 cdCheck(cd);  
 String updateID = (String) session.getAttribute("updateID");  
 cd.setCreateTime(System.currentTimeMillis() / 1000);  
 cd.setUpdateTime(System.currentTimeMillis() / 1000);  
 cd.setCreateBy(updateID);  
 cd.setUpdateBy(updateID);  
 course\_describeRepository.save(cd);  
 return new Course\_describeResp(cd);  
  
 }  
  
 @ApiOperation(value = "删除课程描述信息", notes = "根据用户信息的cd\_id来删除一个课程描述信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "cd\_id", value = "被操作的目标主键,直接放入地址中,替换{cd\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{cd\_id}")  
 public ResponseEntity<Void> deleteCourse(@PathVariable("cd\_id") String cd\_id, HttpSession session) {  
 return course\_describeRepository.findById(cd\_id).map(entity -> {  
 course\_describeRepository.delete(entity);  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 }).orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "更新课程描述信息", notes = "通过cd\_id定位课程描述信息并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "cd\_id", value = "被操作的目标主键,直接放入地址中,替换{cd\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = Course\_describeResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{cd\_id}")  
 public ResponseEntity<Course\_describeResp> updateCourse(@PathVariable("cd\_id") String cd\_id,  
 @ApiParam(value = "需要更新的课程描述信息,以json格式放入Request Body中", required = true) @RequestBody Course\_describeReq course\_describeReq,  
 HttpSession session) {  
  
 Course\_describe newcd = new Course\_describe(course\_describeReq);  
 Course\_describe cd = course\_describeRepository.findByCdID(cd\_id);  
// if (newcd.getCourseID() != null)  
// cd.setCourseID(newcd.getCourseID());  
 if (newcd.getWeekSign() != null)  
 cd.setWeekSign(newcd.getWeekSign());  
 if (newcd.getWeekTime() != null)  
 cd.setWeekTime(newcd.getWeekTime());  
 if (newcd.getCourseBeginTime() != null)  
 cd.setCourseBeginTime(newcd.getCourseBeginTime());  
 if (newcd.getCourseEndTime() != null)  
 cd.setCourseEndTime(newcd.getCourseEndTime());  
 String updateID = (String) session.getAttribute("updateID");  
 cd.setUpdateBy(updateID);  
 cd.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (course\_describeRepository.save(cd) != null)  
 return new ResponseEntity<Course\_describeResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "根据主键查找课程描述信息", notes = "根据课程描述信息cd\_id查找课程描述信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "cd\_id", value = "被操作的目标主键,直接放入地址中,替换{cd\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = Course\_describeResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{cd\_id}")  
 public ResponseEntity<Course\_describeResp> findByID(@PathVariable("cd\_id") String cd\_id) {  
 return course\_describeRepository.findById(cd\_id)  
 .map(entity -> new ResponseEntity<Course\_describeResp>(new Course\_describeResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "大批量增加课程描述信息", notes = "上传课程描述信息表格,以List一次性添加数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("addCourse\_describeByExcel")  
 public List<Course\_describeResp> addCourse\_describeByFile(@RequestParam("file") MultipartFile file,  
 HttpSession session) {  
 if (file == null)  
 return null;  
 String ofname = file.getOriginalFilename();  
 if (ofname == null || ("").equals(ofname) && file.getSize() == 0)  
 return null;  
 ArrayList<Course\_describeResp> cds = new ArrayList<Course\_describeResp>();  
 List<Course\_describeReq> reqList = ExcelFileReader.readfile(file, new Course\_describeReq());  
 for (Course\_describeReq course\_describeReq : reqList) {  
 System.out.println(course\_describeReq.toString());  
 Course\_describe course\_describe = new Course\_describe(course\_describeReq);  
 System.out.println(course\_describe.toString());  
 String updateID = (String) session.getAttribute("updateID");  
 course\_describe.setCreateTime(System.currentTimeMillis() / 1000);  
 course\_describe.setUpdateTime(System.currentTimeMillis() / 1000);  
 course\_describe.setCreateBy(updateID);  
 course\_describe.setUpdateBy(updateID);  
 System.out.println(course\_describe.toString());  
 cds.add(new Course\_describeResp(course\_describe));  
 course\_describeRepository.save(course\_describe);  
 }  
 return cds.isEmpty() ? null : cds;  
 }  
  
 // 课程描述信息自定义完整性校验，有问题就抛异常  
 public void cdCheck(@Valid Course\_describe entity) {  
  
 }  
}

# LoginController.java

package com.isl.controller;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.jboss.logging.Param;  
import org.springframework.http.HttpMethod;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.client.RestTemplate;  
  
import com.google.gson.Gson;  
import com.isl.entity.User;  
import com.isl.entity.request.UserReq;  
import com.isl.entity.response.UserResp;  
import com.isl.service.UserRepository;  
import com.isl.wechat.WeChatSession;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "登录相关接口")  
@RestController  
public class LoginController {  
  
 private UserRepository userRepository;  
  
 public LoginController(UserRepository userRepository) {  
 super();  
 this.userRepository = userRepository;  
 }  
  
 @ApiOperation(value = "登录接口", notes = "通过用户的账号密码登录")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = UserResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/Login")  
 public UserResp Login(  
 @ApiParam(value = "需要相应的用户信息，以json格式放入Request body中", required = true) @Valid @RequestBody UserReq userReq,  
 HttpSession session) {  
 session.setAttribute("updateID", userReq.getRole\_id());  
 String password = userReq.getPassword();  
 System.out.println(userReq.getRole\_id());  
 System.out.println(password);  
 User u = userRepository.findByRoleid(userReq.getRole\_id());  
 System.out.println(u.toString());  
 if (u != null) {  
 if (password.equals(u.getPassword()))// 密码正确，进入其他判断  
 {  
 String RoleIdentity = u.getRoleIdentity();  
 session.setAttribute("userInfo", u);  
 System.out.println("sessionid: " + session.getId());  
 if (RoleIdentity.equals("教师")) {  
 System.out.println("教师登录成功");  
 return new UserResp(u);// 返回用户信息  
 } else if (RoleIdentity.equals("管理员")) {  
 System.out.println("管理员登录成功");  
 return new UserResp(u);// 返回用户信息  
 }  
 } else {  
 return null;// 密码错误返回null  
 }  
 }  
 return null;// 用户不存在返回null  
 }  
  
 @ApiOperation(value = "学生登录接口", notes = "通过code免密登录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "code", value = "学生使用微信号免密登录，直接放入地址中,替换{code}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = UserResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/Login/{code}")  
 public UserResp StuLogin(@PathVariable("code") String code, HttpSession session) {  
 // 通过code调用微信接口获取openID  
 String appId = "wxcac2dd26170e3284";  
 String secret = "feb1520b7e64c22822448003c8b1ae52";  
 String url = "https://api.weixin.qq.com/sns/jscode2session?appid=" + appId + "&secret=" + secret + "&js\_code="  
 + code + "&grant\_type=authorization\_code";  
 RestTemplate restTemplate = new RestTemplate();  
 // 进行网络请求,访问url接口  
 ResponseEntity<String> responseEntity = restTemplate.exchange(url, HttpMethod.GET, null, String.class);  
 // 根据返回值进行后续操作  
 if (responseEntity != null && responseEntity.getStatusCode() == HttpStatus.OK) {  
 String sessionData = responseEntity.getBody();  
 Gson gson = new Gson();  
 // 解析从微信服务器获得的openid和session\_key;  
 WeChatSession weChatSession = gson.fromJson(sessionData, WeChatSession.class);  
 System.out.println(weChatSession);  
 // 获取用户的唯一标识  
 String role\_openID = weChatSession.getOpenid();  
 // 获取会话秘钥  
 String session\_key = weChatSession.getSession\_key();  
 System.out.println("openID: ----" + role\_openID + "\nKey: ----" + session\_key);  
  
 // 学生使用openID免密登录  
 User student = userRepository.findByRoleOpenid(role\_openID);  
 System.out.println(student);  
 if (student != null) {  
 if (!student.getRoleIdentity().equals("学生")) {  
 System.out.println("不是学生，非法免密登录");  
 return null;// 不是学生，非法免密登录  
 }  
 session.setAttribute("updateID", student.getRoleID());  
 session.setAttribute("userInfo", student);  
 System.out.println("sessionid: " + session.getId());  
 System.out.println("学生登录成功");  
 return new UserResp(student);// 返回用户信息  
 }  
 }  
 return null;// 用户不存在返回空  
 }  
  
 @ApiOperation(value = "用户是否登录查询接口", notes = "返回session中的用户信息")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = UserResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/returnUForS")  
 public UserResp returnUserForSession(HttpSession session) {  
 User user = (User) session.getAttribute("userInfo");  
 System.out.println("sessionid: " + session.getId());  
 if (user != null) {  
 System.out.println("返回session中的用户");  
 return new UserResp(user);  
 } else {  
 System.out.println("未返回");  
 return null;  
 }  
 }  
  
 @ApiOperation(value = "登出接口", notes = "删除session中的用户信息")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/LoginOut")  
 public void LoginOut(HttpSession session) {  
  
 session.removeAttribute("userInfo");  
 System.out.println("用户登出");  
 }  
}

# MajorController.java

package com.isl.controller;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
import com.isl.entity.Major;  
import com.isl.entity.request.MajorReq;  
import com.isl.entity.response.MajorResp;  
import com.isl.excel.ExcelFileReader;  
import com.isl.service.MajorRepository;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "专业信息相关接口")  
@RestController  
@RequestMapping("/major")  
public class MajorController {  
  
 private MajorRepository majorRepository;  
  
 /\*\*  
 \* 构造函数注入  
 \*   
 \* @param majorRepository  
 \*/  
 public MajorController(MajorRepository majorRepository) {  
 super();  
 this.majorRepository = majorRepository;  
 }  
  
 @ApiOperation(value = "获取全部专业信息", notes = "获取全部专业信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = MajorResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/allmajor")  
 public List<MajorResp> getAllMajor() {  
 return majorRepository.findAll().stream().map(entity -> new MajorResp(entity)).collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增专业信息", notes = "上传必要的专业信息来创建一条新的专业记录")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = MajorResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/addMajor")  
 public MajorResp addMajor(  
 @ApiParam(value = "需要创建的专业记录，以json格式放入Request body中", required = true) @Valid @RequestBody MajorReq majorReq,  
 HttpSession session) {  
 Major major = new Major(majorReq);  
 Check(major);  
 String updateID = (String) session.getAttribute("updateID");  
 major.setCreateBy(updateID);  
 major.setUpdateBy(updateID);  
 major.setCreateTime(System.currentTimeMillis() / 1000);  
 major.setUpdateTime(System.currentTimeMillis() / 1000);  
 majorRepository.save(major);  
 return new MajorResp(major);  
 }  
  
 @ApiOperation(value = "删除专业记录", notes = "根据专业记录的majorid来删除一个选课记录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "majorid", value = "被操作的目标主键,直接放入地址中,替换{majorid}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{majorid}")  
 public ResponseEntity<Void> deleteMajor(@PathVariable("majorid") String majorid, HttpSession session) {  
 return majorRepository.findById(majorid).map(entity -> {  
 majorRepository.delete(entity);  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 }).orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "更新专业记录", notes = "通过majorid定位选课记录并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "majorid", value = "被操作的目标主键,直接放入地址中,替换{majorid}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = MajorResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{majorid}")  
 public ResponseEntity<MajorResp> updateMajor(@PathVariable("majorid") String majorid,  
 @ApiParam(value = "需要更新的选课记录信息,以json格式放入Request Body中", required = true) @RequestBody MajorReq majorReq,  
 HttpSession session) {  
 Major newmajor = new Major(majorReq);  
 Major major = majorRepository.findByMajorID(majorid);  
 if (newmajor.getMajorName() != null)  
 major.setMajorName(newmajor.getMajorName());  
 if (newmajor.getAcademyID() != null)  
 major.setAcademyID(newmajor.getAcademyID());  
 String updateID = (String) session.getAttribute("updateID");  
 major.setUpdateBy(updateID);  
 major.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (majorRepository.save(major) != null)  
 return new ResponseEntity<MajorResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "根据主键查找专业记录", notes = "根据专业记录主键majorid查找记录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "majorid", value = "被操作的目标主键,直接放入地址中,替换{majorid}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = MajorResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{majorid}")  
 public ResponseEntity<MajorResp> findMajorByID(@PathVariable("majorid") String majorid) {  
 return majorRepository.findById(majorid)  
 .map(entity -> new ResponseEntity<MajorResp>(new MajorResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "大批量增加专业记录信息", notes = "上传专业信息表格,以List一次性添加数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("addMajorByExcel")  
 public List<MajorResp> addMajorByFile(@RequestParam("file") MultipartFile file, HttpSession session) {  
 if (file == null)  
 return null;  
 String ofname = file.getOriginalFilename();  
 if (ofname == null || ("").equals(ofname) && file.getSize() == 0)  
 return null;  
 ArrayList<MajorResp> majorResp = new ArrayList<MajorResp>();  
 List<MajorReq> reqList = ExcelFileReader.readfile(file, new MajorReq());  
 for (MajorReq req : reqList) {  
 System.out.println(req.toString());  
 Major major = new Major(req);  
 System.out.println(major.toString());  
 String updateID = (String) session.getAttribute("updateID");  
 major.setCreateTime(System.currentTimeMillis() / 1000);  
 major.setUpdateTime(System.currentTimeMillis() / 1000);  
 major.setCreateBy(updateID);  
 major.setUpdateBy(updateID);  
 System.out.println(major.toString());  
 majorResp.add(new MajorResp(major));  
 majorRepository.save(major);  
 }  
 return majorResp.isEmpty() ? null : majorResp;  
 }  
  
 public void Check(@Valid Major entity) {  
  
 }  
  
}

# SchoolController.java

package com.isl.controller;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
import com.isl.entity.School;  
import com.isl.entity.request.SchoolReq;  
import com.isl.entity.response.SchoolResp;  
import com.isl.excel.ExcelFileReader;  
import com.isl.service.SchoolRepository;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "学校信息相关接口")  
@RestController  
@RequestMapping("/school")  
public class SchoolController {  
 private SchoolRepository schoolRepository;  
  
 /\*\*  
 \* 构造函数注入  
 \*   
 \* @param schoolRepository  
 \*/  
  
 public SchoolController(SchoolRepository schoolRepository) {  
 this.schoolRepository = schoolRepository;  
 }  
  
 @ApiOperation(value = "获取全部学校信息", notes = "获取全部学校信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = SchoolResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/allSchool")  
 public List<SchoolResp> getAllCourse() {  
 return schoolRepository.findAll()// 获取所有学校信息  
 .stream().map(entity -> new SchoolResp(entity)).collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增学校信息", notes = "上传必要的学校信息来创建一个新的学校")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = SchoolResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/addSchool")  
 public SchoolResp addSchool(  
 @ApiParam(value = "需要创建的学校信息，以json格式放入Request body中", required = true) @Valid @RequestBody SchoolReq schoolReq,  
 HttpSession session) {  
 School school = new School(schoolReq);  
 SchoolCheck(school);  
 String updateID = (String) session.getAttribute("updateID");  
 school.setCreateBy(updateID);  
 school.setUpdateBy(updateID);  
 school.setCreateTime(System.currentTimeMillis() / 1000);  
 school.setUpdateTime(System.currentTimeMillis() / 1000);  
 schoolRepository.save(school);  
 return new SchoolResp(school);  
  
 }  
  
 @ApiOperation(value = "删除学校信息", notes = "根据用户信息的school\_id来删除一个学校信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "school\_id", value = "被操作的目标主键,直接放入地址中,替换{school\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{school\_id}")  
 public ResponseEntity<Void> deleteCourse(@PathVariable("school\_id") String school\_id, HttpSession session) {  
 return schoolRepository.findById(school\_id).map(entity -> {  
 schoolRepository.delete(entity);  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 }).orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "更新学校信息", notes = "通过school\_id定位学校信息并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "school\_id", value = "被操作的目标主键,直接放入地址中,替换{school\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = SchoolResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{school\_id}")  
 public ResponseEntity<SchoolResp> updateSchool(@PathVariable("school\_id") String school\_id,  
 @ApiParam(value = "需要更新的学校信息,以json格式放入Request Body中", required = true) @RequestBody SchoolReq schoolReq,  
 HttpSession session) {  
  
 School newSchool = new School(schoolReq);  
 School school = schoolRepository.findBySchoolID(school\_id);  
 if (newSchool.getSchoolName() != null)  
 school.setSchoolName(newSchool.getSchoolName());  
 if (newSchool.getSchoolAdress() != null)  
 school.setSchoolAdress(newSchool.getSchoolAdress());  
 if (newSchool.getSchoolPostcode() != null)  
 school.setSchoolPostcode(newSchool.getSchoolPostcode());  
 String updateID = (String) session.getAttribute("updateID");  
 school.setUpdateBy(updateID);  
 school.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (schoolRepository.save(school) != null)  
 return new ResponseEntity<SchoolResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
  
 }  
  
 @ApiOperation(value = "根据主键查找学校信息", notes = "根据学校信息school\_id查找学校信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "school\_id", value = "被操作的目标主键,直接放入地址中,替换{school\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = SchoolResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{school\_id}")  
 public ResponseEntity<SchoolResp> findByID(@PathVariable("school\_id") String school\_id) {  
 return schoolRepository.findById(school\_id)  
 .map(entity -> new ResponseEntity<SchoolResp>(new SchoolResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "大批量增加学校信息", notes = "上传学校信息表格,以List一次性添加数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("addSchoolByExcel")  
 public List<SchoolResp> addSchoolByFile(@RequestParam("file") MultipartFile file, HttpSession session) {  
 if (file == null)  
 return null;  
 String ofname = file.getOriginalFilename();  
 if (ofname == null || ("").equals(ofname) && file.getSize() == 0)  
 return null;  
 ArrayList<SchoolResp> schools = new ArrayList<SchoolResp>();  
 List<SchoolReq> reqList = ExcelFileReader.readfile(file, new SchoolReq());  
 for (SchoolReq schoolReq : reqList) {  
 System.out.println(schoolReq.toString());  
 School school = new School(schoolReq);  
 System.out.println(school.toString());  
 String updateID = (String) session.getAttribute("updateID");  
 school.setCreateTime(System.currentTimeMillis() / 1000);  
 school.setUpdateTime(System.currentTimeMillis() / 1000);  
 school.setCreateBy(updateID);  
 school.setUpdateBy(updateID);  
 System.out.println(school.toString());  
 schools.add(new SchoolResp(school));  
 schoolRepository.save(school);  
 }  
 System.out.println(schools);  
 return schools;  
 }  
  
 // 学校信息自定义完整性校验，有问题就抛异常  
 public void SchoolCheck(@Valid School entity) {  
  
 }  
}

# SignPlanController.java

package com.isl.controller;  
  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import com.isl.entity.SignPlan;  
import com.isl.entity.StuSign;  
import com.isl.entity.request.SignPlanReq;  
import com.isl.entity.response.SignPlanResp;  
import com.isl.service.SignPlanRepository;  
import com.isl.service.StuCourseRepository;  
import com.isl.service.StuSignRepository;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "签到计划相关接口")  
@RestController  
@RequestMapping("/signPlan")  
public class SignPlanController {  
 private SignPlanRepository signPlanRepository;  
 private StuCourseRepository stuCourseRepository;  
 private StuSignRepository stuSignRepository;  
  
 /\*\*  
 \* 构造函数注入  
 \*   
 \* @param signPlanRepository  
 \*/  
 public SignPlanController(StuSignRepository stuSignRepository, StuCourseRepository stuCourseRepository,  
 SignPlanRepository signPlanRepository) {  
 super();  
 this.signPlanRepository = signPlanRepository;  
 this.stuCourseRepository = stuCourseRepository;  
 this.stuSignRepository = stuSignRepository;  
 }  
  
 @ApiOperation(value = "获取全部签到安排信息", notes = "获取全部签到安排信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = SignPlanResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/allSp")  
 public List<SignPlanResp> getAllCourse() {  
 return signPlanRepository.findAll().stream().map(entity -> new SignPlanResp(entity))  
 .collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增签到安排信息", notes = "老师通过发起签到来创建一条新的签到安排记录,返回签到安排记录的id")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = String.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/addSp")  
 public String addSp(  
 @ApiParam(value = "需要创建的签到安排记录，以json格式放入Request body中", required = true) @Valid @RequestBody SignPlanReq signplanReq,  
 HttpSession session) {  
 SignPlan signPlan = new SignPlan(signplanReq);  
 Check(signPlan);  
 String updateID = (String) session.getAttribute("updateID");  
 signPlan.setCreateBy(updateID);  
 signPlan.setUpdateBy(updateID);  
 signPlan.setBeginTime(String.valueOf(System.currentTimeMillis() / 1000));// 当前时间即为签到开始时间  
 signPlan.setEndTime(String.valueOf((System.currentTimeMillis() / 1000) + signPlan.getSustainTime() \* 60));// 当前时间加持续时间为结束时间  
 long endTime = (System.currentTimeMillis() / 1000) + signPlan.getSustainTime() \* 60;  
 session.setAttribute("signPlanTime", endTime);  
 signPlan.setCreateTime(System.currentTimeMillis() / 1000);  
 signPlan.setUpdateTime(System.currentTimeMillis() / 1000);  
 signPlan.setSignStatus("正在正常签到");  
 signPlanRepository.save(signPlan);  
  
 /\*\*  
 \* 发起签到后同时将所有选课学生导入StuSign, 并且签到状态设为0(未签到)  
 \*\*/  
 List<String> stuids = stuCourseRepository.findStuByCourseID(signPlan.getCourseID());  
 for (String stuid : stuids) {  
 StuSign stuSign = new StuSign();  
 stuSign.setStudentID(stuid);  
 stuSign.setSignPlanID(signPlan.getSignPlanID());  
 stuSign.setStusignState(0);  
 stuSignRepository.save(stuSign);  
 }  
   
 return signPlan.getSignPlanID();  
 // session.setAttribute("SignPlanId", signPlan.getSignPlanID());  
//   
// try {  
// Thread.sleep(signPlan.getSustainTime()\*60000);//线程等待时间为持续时间  
// } catch (InterruptedException e) {  
// e.printStackTrace();  
// }   
// signPlan.setSignStatus("签到已结束");  
// signPlanRepository.save(signPlan);  
 }  
  
 @ApiOperation(value = "删除签到安排记录", notes = "根据签到安排信息的sp\_id来删除一个签到安排记录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "sp\_id", value = "被操作的目标主键,直接放入地址中,替换{sp\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{sp\_id}")  
 public ResponseEntity<Void> deleteSp(@PathVariable("sp\_id") String sp\_id, HttpSession session) {  
 SignPlan sp = signPlanRepository.findBySpID(sp\_id);  
 String updateID = (String) session.getAttribute("updateID");  
 sp.setDelete\_state(1);  
 sp.setUpdateBy(updateID);  
 sp.setUpdateTime(System.currentTimeMillis() / 1000);  
 if ((signPlanRepository.save(sp)) != null)  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "更新签到安排记录", notes = "通过sp\_id定位签到安排记录并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "sp\_id", value = "被操作的目标主键,直接放入地址中,替换{sp\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = SignPlanResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{sp\_id}")  
 public ResponseEntity<SignPlanResp> updateSp(@PathVariable("sp\_id") String sp\_id,  
 @ApiParam(value = "需要更新的签到安排记录信息,以json格式放入Request Body中", required = true) @RequestBody SignPlanReq signPlanReq,  
 HttpSession session) {  
  
 SignPlan newsp = new SignPlan(signPlanReq);  
 SignPlan sp = signPlanRepository.findBySpID(sp\_id);  
 if (newsp.getCourseID() != null)  
 sp.setCourseID(newsp.getCourseID());  
 if (String.valueOf(newsp.getSustainTime()) != null)  
 sp.setSustainTime(newsp.getSustainTime());  
 String updateID = (String) session.getAttribute("updateID");  
 sp.setUpdateBy(updateID);  
 sp.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (signPlanRepository.save(sp) != null)  
 return new ResponseEntity<SignPlanResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "根据主键查找签到安排记录", notes = "根据签到安排记录主键sp\_id查找记录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "sp\_id", value = "被操作的目标主键,直接放入地址中,替换{sp\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = SignPlanResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{sp\_id}")  
 public ResponseEntity<SignPlanResp> findByID(@PathVariable("sp\_id") String sp\_id) {  
 return signPlanRepository.findById(sp\_id)  
 .map(entity -> new ResponseEntity<SignPlanResp>(new SignPlanResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "根据签到状态返回密文", notes = "直接获取对应密文")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/getCiphertext")  
 public String returnPTxt(@RequestBody String SignPlanId) {  
 String SignStatus = "正在正常签到";  
 SignPlan signPlan = new SignPlan();  
// System.out.println(session.getId());  
// String SignPlanId=(String) session.getAttribute("SignPlanId");  
 System.out.println("signplanid:" + SignPlanId);  
 signPlan = signPlanRepository.findBySpID(SignPlanId);  
 System.out.println(signPlan.toString());  
 String Ciphertext = String.valueOf(System.currentTimeMillis() / 1000);  
 if (SignStatus.equals(signPlan.getSignStatus())) {  
 signPlan.setCiphertext(Ciphertext);  
 this.signPlanRepository.save(signPlan);  
 return Ciphertext;// 处于签到期间，返回密文  
 } else {  
 return " 签到已结束";// 签到结束，返回空  
 }  
 }  
  
 // 签到安排信息校验  
 public void Check(@Valid SignPlan entity) {  
  
 }  
  
}

# StuCourseController.java

package com.isl.controller;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.multipart.MultipartFile;  
  
import com.isl.entity.StuCourse;  
import com.isl.entity.Student;  
import com.isl.entity.request.StuCourseReq;  
import com.isl.entity.response.StuCourseResp;  
import com.isl.excel.ExcelFileReader;  
import com.isl.service.StuCourseRepository;  
import com.isl.service.StudentRepository;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "学生选课信息相关接口")  
@RestController  
@RequestMapping("/stuCourse")  
public class StuCourseController {  
 private StuCourseRepository stuCourseRepository;  
 private StudentRepository studentRepository;  
  
 /\*\*  
 \* 构造函数注入  
 \*/  
 public StuCourseController(StuCourseRepository stuCourseRepository, StudentRepository studentRepository) {  
 super();  
 this.stuCourseRepository = stuCourseRepository;  
 this.studentRepository = studentRepository;  
 }  
  
 @ApiOperation(value = "获取全部选课信息", notes = "获取全部选课信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = StuCourseResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/allSc")  
 public List<StuCourseResp> getAllSc() {  
 return stuCourseRepository.findAll().stream().map(entity -> new StuCourseResp(entity))  
 .collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增选课信息", notes = "上传必要的选课信息来创建一条新的选课记录")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = StuCourseResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/addSc")  
 public StuCourseResp addSc(  
 @ApiParam(value = "需要创建的选课记录，以json格式放入Request body中", required = true) @Valid @RequestBody StuCourseReq stuCourseReq,  
 HttpSession session) {  
 StuCourse sc = new StuCourse(stuCourseReq);  
 Check(sc);  
 String updateID = (String) session.getAttribute("updateID");  
 sc.setCreateBy(updateID);  
 sc.setUpdateBy(updateID);  
 sc.setCreateTime(System.currentTimeMillis() / 1000);  
 sc.setUpdateTime(System.currentTimeMillis() / 1000);  
 stuCourseRepository.save(sc);  
 return new StuCourseResp(sc);  
 }  
  
 @ApiOperation(value = "删除选课记录", notes = "根据选课记录的sc\_id来删除一个选课记录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "sc\_id", value = "被操作的目标主键,直接放入地址中,替换{sc\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{sc\_id}")  
 public ResponseEntity<Void> deleteSc(@PathVariable("sc\_id") String sc\_id, HttpSession session) {  
 StuCourse sc = stuCourseRepository.findByScID(sc\_id);  
 sc.setDeleteState(1);  
 String updateID = (String) session.getAttribute("updateID");  
 sc.setUpdateBy(updateID);  
 sc.setUpdateTime(System.currentTimeMillis() / 1000);  
 if ((stuCourseRepository.save(sc)) != null)  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "更新选课记录", notes = "通过sc\_id定位选课记录并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "sc\_id", value = "被操作的目标主键,直接放入地址中,替换{sc\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = StuCourseResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{sc\_id}")  
 public ResponseEntity<StuCourseResp> updateSc(@PathVariable("sc\_id") String sc\_id,  
 @ApiParam(value = "需要更新的选课记录信息,以json格式放入Request Body中", required = true) @RequestBody StuCourseReq stuCourseReq,  
 HttpSession session) {  
 StuCourse newsc = new StuCourse(stuCourseReq);  
 StuCourse sc = stuCourseRepository.findByScID(sc\_id);  
 if (newsc.getCourseID() != null)  
 sc.setCourseID(newsc.getCourseID());  
 if (newsc.getStudentID() != null)  
 sc.setStudentID(newsc.getStudentID());  
 String updateID = (String) session.getAttribute("updateID");  
 sc.setUpdateBy(updateID);  
 sc.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (stuCourseRepository.save(sc) != null)  
 return new ResponseEntity<StuCourseResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "根据主键查找选课记录", notes = "根据选课记录主键sc\_id查找记录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "sc\_id", value = "被操作的目标主键,直接放入地址中,替换{sc\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = StuCourseResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{sc\_id}")  
 public ResponseEntity<StuCourseResp> findByID(@PathVariable("sc\_id") String sc\_id) {  
 return stuCourseRepository.findById(sc\_id)  
 .map(entity -> new ResponseEntity<StuCourseResp>(new StuCourseResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "大批量增加学生选课信息", notes = "上传学生选课信息表格,以List一次性添加数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("addSCByExcel")  
 public List<StuCourseResp> addSCByFile(@RequestParam("file") MultipartFile file, HttpSession session) {  
 if (file == null)  
 return null;  
 String ofname = file.getOriginalFilename();  
 if (ofname == null || ("").equals(ofname) && file.getSize() == 0)  
 return null;  
 ArrayList<StuCourseResp> stuCourses = new ArrayList<StuCourseResp>();  
 List<StuCourseReq> reqList = ExcelFileReader.readfile(file, new StuCourseReq());  
 for (StuCourseReq screq : reqList) {  
 System.out.println(screq.toString());  
 StuCourse sc = new StuCourse(screq);  
 System.out.println(sc.toString());  
 String updateID = (String) session.getAttribute("updateID");  
 sc.setCreateTime(System.currentTimeMillis() / 1000);  
 sc.setUpdateTime(System.currentTimeMillis() / 1000);  
 sc.setCreateBy(updateID);  
 sc.setUpdateBy(updateID);  
 System.out.println(sc.toString());  
 stuCourses.add(new StuCourseResp(sc));  
 stuCourseRepository.save(sc);  
 }  
 return stuCourses;  
 }  
  
 @ApiOperation(value = "获取该课程全部选课学生信息", notes = "获取该课程全部选课学生信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/allSByCid")  
 public List<Student> getAllStudentByCourseid(  
 @ApiParam(value = "需要查询的课程id，放入Request body中", required = true) @RequestBody String course\_id) {  
 return studentRepository.findAllStudents(course\_id).stream().collect(Collectors.toList());  
 /\*  
 \* System.out.println(course\_id); List<Student>  
 \* list=studentRepository.findAllStudents(course\_id); System.out.println(list);  
 \* return null;  
 \*/  
 }  
  
 public void Check(@Valid StuCourse entity) {  
  
 }  
}

# StuSignController.java

package com.isl.controller;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
  
import com.isl.entity.SignPlan;  
import com.isl.entity.StuCourse;  
import com.isl.entity.StuSign;  
import com.isl.entity.request.StuSignReq;  
import com.isl.entity.response.StuSignResp;  
import com.isl.entity.response.StuSignStatusResp;  
import com.isl.service.SignPlanRepository;  
import com.isl.service.StuCourseRepository;  
import com.isl.service.StuSignRepository;  
import com.isl.service.UserRepository;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "学生签到信息相关接口")  
@RestController  
@RequestMapping("/stuSign")  
public class StuSignController {  
  
 private StuSignRepository stuSignRepository;  
 private SignPlanRepository signPlanRepository;  
 private StuCourseRepository stuCourseRepository;  
 private UserRepository userRepository;  
  
 /\*\*  
 \* 构造函数注入  
 \*/  
 public StuSignController(UserRepository userRepository, StuCourseRepository stuCourseRepository,  
 SignPlanRepository signPlanRepository, StuSignRepository stuSignRepository) {  
 this.stuSignRepository = stuSignRepository;  
 this.signPlanRepository = signPlanRepository;  
 this.stuCourseRepository = stuCourseRepository;  
 this.userRepository = userRepository;  
 }  
  
 @ApiOperation(value = "获取全部签到信息", notes = "获取全部签到信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = StuSignResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/allSs")  
 public List<StuSignResp> getAllSc() {  
 return stuSignRepository.findAll().stream().map(entity -> new StuSignResp(entity)).collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增签到信息", notes = "学生通过扫码来上传自身的信息从而创建一条新的签到记录")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = StuSignResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/addSs")  
 public StuSignResp addSc(  
 @ApiParam(value = "需要创建的签到记录，以json格式放入Request body中", required = true) @Valid @RequestBody StuSignReq stuSignReq,  
 HttpSession session) {  
 //System.out.println("1");  
 StuSign ss = new StuSign(stuSignReq);  
 String studentID=this.userRepository.findRoleidByOpenid(ss.getOpenID());  
 ss.setStudentID(studentID);  
 //System.out.println(2);  
 SignPlan sp=this.signPlanRepository.findBySpID(ss.getSignPlanID());  
 //System.out.println(3);  
 // System.out.println("1");  
 // 通过修改(因为记录已经在发起签到时保存)  
 // System.out.println(2);  
 // System.out.println(3);  
 Check(ss);  
 /\*\* 检验 \*\*/  
 String courseID = sp.getCourseID();  
 StuCourse haveStuCourse = this.stuCourseRepository.findBySCID(ss.getStudentID(), courseID);  
 if (haveStuCourse == null) {  
 System.out.println("学生未选课");  
 return null;  
 }  
 String Ciphertext = sp.getCiphertext();  
 System.out.println("扫码:---" + ss.getCiphertext() + "\n二维码生成：----- " + Ciphertext);  
 if ((Long.valueOf(ss.getCiphertext()) - Long.valueOf(Ciphertext)) < 10) {  
 // 截止时间  
 long tea\_signTime = Long.valueOf(signPlanRepository.findBySpID(ss.getSignPlanID()).getEndTime());  
 // 当前时间  
 long stu\_signTime = System.currentTimeMillis() / 1000 + 1;  
  
 if (tea\_signTime - stu\_signTime > 0)// 对当前时间和签到截止时间进行判断  
 {  
 ss.setStusignState(1);// 正常签到  
 } else {  
 ss.setStusignState(2);// 迟到  
 }  
  
 // stuSignReq.setSignPlanID(signPlanID);  
 String updateID = (String) session.getAttribute("updateID");  
 ss.setUpdateBy(updateID);  
 ss.setCreateBy(updateID);  
 ss.setUpdateTime(System.currentTimeMillis() / 1000);  
 ss.setCreateTime(System.currentTimeMillis() / 1000);  
 stuSignRepository.save(ss);  
 System.out.println("签到成功");  
 return new StuSignResp(ss); // 签到成功返回信息  
 } else {  
 return null;// 签到失败，二维码过期  
 }  
  
 }  
  
 @ApiOperation(value = "删除签到记录", notes = "根据签到记录的ss\_id来删除一个签到记录")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "ss\_id", value = "被操作的目标主键,直接放入地址中,替换{ss\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{ss\_id}")  
 public ResponseEntity<Void> deleteSc(@PathVariable("ss\_id") String ss\_id, HttpSession session) {  
 return stuSignRepository.findById(ss\_id).map(entity -> {  
 stuSignRepository.delete(entity);  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 }).orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "更新签到记录信息", notes = "通过ss\_id定位签到记录信息并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "ss\_id", value = "被操作的目标主键,直接放入地址中,替换{ss\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = StuSignResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{ss\_id}")  
 public ResponseEntity<StuSignResp> updateCourse(@PathVariable("ss\_id") String ss\_id,  
 @ApiParam(value = "需要更新的签到记录信息,以json格式放入Request Body中", required = true) @RequestBody StuSignReq stuSignReq,  
 HttpSession session) {  
  
 StuSign newss = new StuSign(stuSignReq);  
 StuSign ss = stuSignRepository.findBySsID(ss\_id);  
 if (newss.getStudentID() != null)  
 ss.setStudentID(newss.getStudentID());  
 if (newss.getSignPlanID() != null)  
 ss.setStuSignID(newss.getStuSignID());  
 if (String.valueOf(newss.getStusignState()) != null)  
 ss.setStusignState(newss.getStusignState());  
 String updateID = (String) session.getAttribute("updateID");  
 ss.setUpdateBy(updateID);  
 ss.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (stuSignRepository.save(ss) != null)  
 return new ResponseEntity<StuSignResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
  
 }  
  
 @ApiOperation(value = "根据主键查找签到记录信息", notes = "根据签到记录信息course\_id查找签到记录信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "ss\_id", value = "被操作的目标主键,直接放入地址中,替换{ss\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = StuSignResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{ss\_id}")  
 public ResponseEntity<StuSignResp> findByID(@PathVariable("ss\_id") String ss\_id) {  
 return stuSignRepository.findById(ss\_id)  
 .map(entity -> new ResponseEntity<StuSignResp>(new StuSignResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "根据sign\_planid查询所有学生签到状况", notes = "根据签到记录信息course\_id查找签到记录信息")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = StuSignResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/getStuSignStatus")  
 public List<StuSignStatusResp> getStuSignStatus(@RequestBody String sign\_planid) {  
 List<StuSign> stuSigns = stuSignRepository.findBySignPlanID(sign\_planid);  
 ArrayList<StuSignStatusResp> stuSignStatusResps = new ArrayList<StuSignStatusResp>();  
 for (StuSign stuSign : stuSigns) {  
 stuSignStatusResps.add(new StuSignStatusResp(userRepository.findByRoleid(stuSign.getStudentID()), stuSign));  
 }  
 return stuSignStatusResps.isEmpty() ? null : stuSignStatusResps;  
 }  
  
 public void Check(@Valid StuSign entity) {  
  
 }  
}

# UserController.java

package com.isl.controller;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import javax.servlet.http.HttpSession;  
import javax.validation.Valid;  
  
import org.springframework.http.HttpMethod;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.client.RestTemplate;  
import org.springframework.web.multipart.MultipartFile;  
  
import com.google.gson.Gson;  
import com.isl.entity.User;  
import com.isl.entity.request.UserReq;  
import com.isl.entity.response.UserResp;  
import com.isl.excel.ExcelFileReader;  
import com.isl.service.UserRepository;  
import com.isl.wechat.WeChatSession;  
  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiImplicitParam;  
import io.swagger.annotations.ApiImplicitParams;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiParam;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
  
@Api(tags = "用户信息相关接口")  
@RestController  
@RequestMapping("/User")  
public class UserController {  
 private UserRepository userRepository;  
  
 /\*\*  
 \* 构造函数注入  
 \*   
 \* @param userRepository  
 \*/  
  
 public UserController(UserRepository userRepository) {  
 super();  
 this.userRepository = userRepository;  
 }  
  
 @ApiOperation(value = "获取全部用户信息", notes = "获取全部用户信息,以数组形式一次性返回数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = UserResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/allUser")  
 public List<UserResp> getAllUser() {  
 return userRepository.findAll()// 获取所有的用户信息  
 .stream().map(entity -> new UserResp(entity))// 根据java8流编程，将User实体转成UserResp实体，  
 .collect(Collectors.toList());  
 }  
  
 @ApiOperation(value = "新增用户信息", notes = "上传必要的用户信息来创建一个新的用户")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = UserResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("/addUser")  
 public UserResp addUser(  
 @ApiParam(value = "需要创建的用户信息,以json格式放入Request Body中", required = true) @Valid @RequestBody UserReq userReq,  
 HttpSession session) {  
  
 User user = new User(userReq);  
 System.out.println(user.toString());  
 UserCheck(user);  
 String updateID = (String) session.getAttribute("updateID");  
 user.setCreateTime(System.currentTimeMillis() / 1000);  
 user.setUpdateTime(System.currentTimeMillis() / 1000);  
 user.setCreateBy(updateID);  
 user.setUpdateBy(updateID);  
 userRepository.save(user);  
 return new UserResp(user);  
 }  
  
 @ApiOperation(value = "删除用户信息", notes = "根据用户信息的user\_id来删除一个用户信息（将删除标志置为1）")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "user\_id", value = "被操作的目标主键,直接放入地址中,替换{user\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @DeleteMapping("/{user\_id}")  
 public ResponseEntity<Void> deleteUser(@PathVariable("user\_id") String user\_id, HttpSession session) {  
 User user = userRepository.findByUserID(user\_id);  
 user.setDeleteState(1);  
 String updateID = (String) session.getAttribute("updateID");  
 user.setUpdateBy(updateID);  
 user.setUpdateTime(System.currentTimeMillis() / 1000);  
 if ((userRepository.save(user)) != null)  
 return new ResponseEntity<Void>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "更新用户信息信息", notes = "通过user\_id定位用户信息并更新其信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "user\_id", value = "被操作的目标主键,直接放入地址中,替换{user\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = UserResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/{user\_id}")  
 public ResponseEntity<UserResp> updateUser(@PathVariable("user\_id") String user\_id,  
 @ApiParam(value = "需要更新的用户信息,以json格式放入Request Body中", required = true) @RequestBody UserReq userReq,  
 HttpSession session) {  
  
 User newUser = new User(userReq);  
 User user = userRepository.findByUserID(user\_id);  
 if (newUser.getRoleID() != null)  
 user.setRoleID(newUser.getRoleID());  
 if (newUser.getPassword() != null)  
 user.setPassword(newUser.getPassword());  
 if (newUser.getRoleName() != null)  
 user.setRoleName(newUser.getRoleName());  
 if (newUser.getRoleIdentity() != null)  
 user.setRoleIdentity(newUser.getRoleIdentity());  
 if (newUser.getRoleOpenID() != null)  
 user.setRoleOpenID(newUser.getRoleOpenID());  
 if (String.valueOf(newUser.getClassID()) != null)  
 user.setClassID(newUser.getClassID());  
 String updateID = (String) session.getAttribute("updateID");  
 user.setUpdateBy(updateID);  
 user.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (userRepository.save(user) != null)  
 return new ResponseEntity<UserResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "根据主键查找用户信息", notes = "根据用户主键user\_id查找学院信息")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "user\_id", value = "被操作的目标主键,直接放入地址中,替换{user\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = UserResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/{user\_id}")  
 public ResponseEntity<UserResp> findUserByID(@PathVariable("user\_id") String user\_id) {  
 return userRepository.findById(user\_id)  
 .map(entity -> new ResponseEntity<UserResp>(new UserResp(entity), HttpStatus.OK))  
 .orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));  
 }  
  
 @ApiOperation(value = "大批量增加用户信息", notes = "上传用户信息表格,以List一次性添加数据")  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功"), @ApiResponse(code = 500, message = "服务器内部异常"),  
 @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"), @ApiResponse(code = 405, message = "权限不足") })  
 @PostMapping("addUserByExcel")  
 public List<UserResp> addUserByFile(@RequestParam("file") MultipartFile file, HttpSession session) {  
 if (file == null)  
 return null;  
 String ofname = file.getOriginalFilename();  
 if (ofname == null || ("").equals(ofname) && file.getSize() == 0)  
 return null;  
 List<UserReq> userreqs = ExcelFileReader.readfile(file, new UserReq());  
 List<UserResp> users = new ArrayList<UserResp>();  
 for (UserReq userreq : userreqs) {  
 User user = new User(userreq);  
 String updateID = (String) session.getAttribute("updateID");  
 user.setCreateBy(updateID);  
 user.setUpdateBy(updateID);  
 user.setCreateTime(System.currentTimeMillis() / 1000);  
 user.setUpdateTime(System.currentTimeMillis() / 1000);  
 user.setDeleteState(0);  
 userRepository.save(user);  
 users.add(new UserResp(user));  
 }  
 return users.isEmpty() ? null : users;  
 }  
  
 @ApiOperation(value = "绑定微信id", notes = "通过open\_id设置用户的微信id")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "open\_id", value = "被操作的目标主键,直接放入地址中,替换{open\_id}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = UserResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @PutMapping("/uWithOpenid/{open\_id}")  
 public ResponseEntity<UserResp> SetRoleID(@PathVariable("open\_id") String open\_id,  
 @ApiParam(value = "需要更新的用户信息,以json格式放入Request Body中", required = true) @RequestBody UserReq userReq,  
 HttpSession session) {  
  
 User user = userRepository.findByRoleid(userReq.getRole\_id());// 通过学号获取该用户信息  
 user.setRoleOpenID(open\_id);  
 String updateID = (String) session.getAttribute("updateID");  
 user.setUpdateBy(updateID);  
 user.setUpdateTime(System.currentTimeMillis() / 1000);  
 if (userRepository.save(user) != null)  
 return new ResponseEntity<UserResp>(HttpStatus.OK);  
 else  
 return new ResponseEntity<>(HttpStatus.NOT\_FOUND);  
 }  
  
 @ApiOperation(value = "获取用户openID接口", notes = "通过code获取openID")  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = "path", name = "code", value = "直接放入地址中,替换{code}", required = true) })  
 @ApiResponses({ @ApiResponse(code = 200, message = "操作成功", response = UserResp.class),  
 @ApiResponse(code = 500, message = "服务器内部异常"), @ApiResponse(code = 400, message = "客户端请求的语法错误,服务器无法理解"),  
 @ApiResponse(code = 405, message = "权限不足") })  
 @GetMapping("/getOpenID/{code}")  
 public String getOpenID(@PathVariable("code") String code, HttpSession session) {  
 // 通过code调用微信接口获取openID  
 String appId = "wxcac2dd26170e3284";  
 String secret = "feb1520b7e64c22822448003c8b1ae52";  
 String url = "https://api.weixin.qq.com/sns/jscode2session?appid=" + appId + "&secret=" + secret + "&js\_code="  
 + code + "&grant\_type=authorization\_code";  
 RestTemplate restTemplate = new RestTemplate();  
 // 进行网络请求,访问url接口  
 ResponseEntity<String> responseEntity = restTemplate.exchange(url, HttpMethod.GET, null, String.class);  
 // 根据返回值进行后续操作  
 if (responseEntity != null && responseEntity.getStatusCode() == HttpStatus.OK) {  
 String sessionData = responseEntity.getBody();  
 Gson gson = new Gson();  
 // 解析从微信服务器获得的openid和session\_key;  
 WeChatSession weChatSession = gson.fromJson(sessionData, WeChatSession.class);  
 System.out.println(weChatSession);  
 // 获取用户的唯一标识  
 String role\_openID = weChatSession.getOpenid();  
 // 获取会话秘钥  
 String session\_key = weChatSession.getSession\_key();  
 System.out.println("openID: ----" + role\_openID + "\nKey: ----" + session\_key);  
 session.setAttribute("session\_key", session\_key);  
 return role\_openID;  
 }  
 return null;  
 }  
  
 // 用户信息自定义完整性校验，有问题就抛异常  
 public void UserCheck(@Valid User entity) {  
  
 }  
  
}

# Academy.java

package com.isl.entity;  
  
import java.util.Date;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
import org.springframework.format.annotation.DateTimeFormat;  
  
import com.isl.entity.request.AcademyReq;  
  
import lombok.Data;  
  
  
  
/\*\*  
 \* 学院信息实体类  
 \* @author Jason Chiang  
 \*  
 \*/  
  
@Entity  
@Table(name="academy")  
@Data  
public class Academy {  
   
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")  
 private String academyID;  
   
 /\*\* 学院名称 \*\*/  
 private String academyName;  
   
 /\*\*学校编号\*\*/  
 private String schoolID;  
   
 /\*\* 创建者 \*\*/  
 private String createBy;  
   
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
   
 /\*\* 创建时间 \*\*/  
 @DateTimeFormat(pattern="yyyy-MM-dd")  
 private Date createTime;  
  
 /\*\* 更新时间 \*\*/  
 @DateTimeFormat(pattern="yyyy-MM-dd")  
 private Date updateTime;  
   
   
   
 public Academy() {  
 super();  
 }  
   
 public Academy(AcademyReq req) {  
 super();  
 this.academyName = req.getAcademy\_name();  
 this.schoolID=req.getSchoolID();  
 }  
   
   
}

# Class.java

package com.isl.entity;  
  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
  
import com.isl.entity.request.ClassReq;  
  
import lombok.Data;  
  
  
  
/\*\*  
 \* 班级信息实体类  
 \* @author 17678  
 \*  
 \*/  
@Entity  
@Table(name="Class")  
@Data  
public class Class {  
  
   
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")  
 private String classID;//班级编号  
   
 /\*\*班级名称\*\*/  
 private String className;//比如计科174  
   
 /\*\*专业编号\*\*/  
 private String majorID;  
   
 /\*\* 创建者 \*\*/  
 private String createBy;  
   
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
   
 /\*\*创建时间（时间戳）\*\*/  
 private Long createTime;  
   
 /\*\*更新时间\*\*/  
 private Long updateTime;  
   
 public Class()  
 {  
 super();  
 }  
 public Class(ClassReq req)  
 {  
 super();  
 this.className=req.getClassName();  
 this.majorID=req.getMajorID();  
 }  
}

# Course.java

package com.isl.entity;  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
  
import com.isl.entity.request.CourseReq;  
  
import lombok.Data;  
  
/\*\*  
 \* 课程信息实体类  
 \*   
 \* @author 17678  
 \*  
 \*/  
@Entity  
@Table(name = "course")  
@Data  
public class Course {  
  
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")  
 private String courseID;  
  
 /\*\* 老师编号 \*\*/  
 private String teacherID;  
  
 /\*\* 管理员编号 \*\*/  
 private String administratorID;  
  
 /\*\* 课程描述编号 \*\*/  
 private String describeID;  
  
 /\*\* 专业名 \*\*/  
 private String majorName;  
  
 /\*\* 课程名 \*\*/  
 private String courseName;  
  
 /\*\* 第一周开始时间 \*\*/  
 private String beginTime;  
  
 /\*\* 学时 \*\*/  
 private int hours;  
  
 /\*\* 地点 \*\*/  
 private String courseLocation;  
  
 /\*\* 创建者 \*\*/  
 private String createBy;  
  
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
  
 /\*\* 创建时间（时间戳） \*\*/  
 private Long createTime;  
  
 /\*\* 更新时间 \*\*/  
 private Long updateTime;  
  
 public Course() {  
 super();  
 }  
  
 public Course(CourseReq req) {  
 this.describeID = req.getDescribeID();  
 this.teacherID = req.getTeacherID();  
 this.administratorID = req.getAdministratorID();  
 this.majorName = req.getMajorName();  
 this.courseName = req.getCourseName();  
 this.beginTime = req.getBeginTime();  
 this.hours = req.getHours();  
 this.courseLocation = req.getCourseLocation();  
 }  
}

# Course\_describe.java

package com.isl.entity;  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
  
import com.isl.entity.request.Course\_describeReq;  
  
import lombok.Data;  
  
  
/\*\*  
 \* 课程描述类  
 \* @author 17678  
 \*  
 \*/  
@Entity  
@Table(name="course\_describe")  
@Data  
public class Course\_describe {  
  
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")//jpa自动生成uuid  
 private String describeID;  
   
// /\*\*课程编号\*\*/  
// private String courseID;  
   
 /\*\*单双周标识\*\*/  
 private String weekSign;  
   
 /\*\*周几\*\*/  
 private String weekTime;  
   
 /\*\*课程开始时间\*\*/  
 private String courseBeginTime;  
   
 /\*\*课程结束时间\*\*/  
 private String courseEndTime;  
   
 /\*\* 创建者 \*\*/  
 private String createBy;  
   
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
   
 /\*\*创建时间（时间戳）\*\*/  
 private Long createTime;  
   
 /\*\*更新时间\*\*/  
 private Long updateTime;  
   
 public Course\_describe()  
 {  
 super();  
 }  
   
 public Course\_describe(Course\_describeReq req)  
 {  
// this.courseID=req.getCourseID();  
 this.weekSign=req.getWeekSign();  
 this.weekTime=req.getWeekTime();  
 this.courseBeginTime=req.getCourseBeginTime();  
 this.courseEndTime=req.getCourseEndTime();  
 }  
}

# Major.java

package com.isl.entity;  
  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
  
import com.isl.entity.request.MajorReq;  
  
import lombok.Data;  
  
  
  
/\*\*  
 \* 专业信息实体类  
 \* @author 176783  
 \*  
 \*/  
  
@Entity  
@Table(name="major")  
@Data  
public class Major {  
  
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")  
 private String majorID;//专业编号  
   
 /\*\*专业名称\*\*/  
 private String majorName;  
   
 /\*\*学院编号\*\*/  
 private String academyID;  
   
   
 /\*\* 创建者 \*\*/  
 private String createBy;  
   
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
   
 /\*\*创建时间（时间戳）\*\*/  
 private Long createTime;  
   
 /\*\*更新时间\*\*/  
 private Long updateTime;  
   
 public Major()  
 {  
 super();  
 }  
 public Major(MajorReq req)  
 {  
 super();  
 this.academyID=req.getAcademyID();  
 this.majorName=req.getMajorName();  
 }  
}

# School.java

package com.isl.entity;  
  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
  
import com.isl.entity.request.SchoolReq;  
  
import lombok.Data;  
  
  
  
/\*\*  
 \* 学校信息实体类  
 \* @author 17678  
 \*  
 \*/  
@Entity  
@Table(name="school")  
@Data  
public class School {  
   
   
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")  
 private String schoolID;  
   
 /\*\*学校名称\*\*/  
 private String schoolName;  
   
 /\*\*学校地址\*\*/  
 private String schoolAdress;  
   
 /\*\*学校邮编\*\*/  
 private String schoolPostcode;  
   
 /\*\* 创建者 \*\*/  
 private String createBy;  
   
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
   
 /\*\*创建时间（时间戳）\*\*/  
 private Long createTime;  
   
 /\*\*更新时间\*\*/  
 private Long updateTime;  
   
 public School()  
 {  
 super();  
 }  
 public School(SchoolReq req)  
 {  
 super();  
 this.schoolName=req.getSchool\_name();  
 this.schoolAdress=req.getSchool\_adress();  
 this.schoolPostcode=req.getSchool\_postcode();  
 }  
}

# SignPlan.java

package com.isl.entity;  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
  
import com.isl.entity.request.SignPlanReq;  
  
import lombok.Data;  
  
  
  
@Entity  
@Table(name="SignPlan")  
@Data  
public class SignPlan {  
  
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")  
 private String signPlanID;  
   
 /\*\* 课程编号 \*\*/  
 private String courseID;  
   
 /\*\* 签到开始时间 \*\*/  
 private String beginTime;  
   
 /\*\* 签到结束时间 \*\*/  
 private String endTime;  
   
 /\*\* 签到持续时间 \*\*/  
 private int sustainTime;  
   
 /\*\*签到时，二维码的密文\*\*/  
 private String Ciphertext;  
   
 /\*\* 签到状态 \*\*/  
 private String signStatus;  
   
 /\*\* 删除标志 \*\*/  
 private int delete\_state;  
   
 /\*\* 创建者 \*\*/  
 private String createBy;  
   
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
   
 /\*\*创建时间（时间戳）\*\*/  
 private Long createTime;  
   
 /\*\*更新时间\*\*/  
 private Long updateTime;  
   
 public SignPlan(SignPlanReq req)  
 {  
 super();  
 this.courseID=req.getCourseID();  
 this.sustainTime=req.getSustainTime();  
   
 }  
  
 @Override  
 public String toString() {  
 return "SignPlan [signPlanID=" + signPlanID + ", courseID=" + courseID + ", beginTime=" + beginTime  
 + ", endTime=" + endTime + ", sustainTime=" + sustainTime + ", signStatus=" + signStatus  
 + ", delete\_state=" + delete\_state + ", createBy=" + createBy + ", updateBy=" + updateBy  
 + ", createTime=" + createTime + ", updateTime=" + updateTime + "]";  
 }  
  
 public SignPlan() {  
 super();  
 // TODO Auto-generated constructor stub  
 }  
   
}

# StuCourse.java

package com.isl.entity;  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
  
import com.isl.entity.request.StuCourseReq;  
  
import lombok.Data;  
  
  
  
@Entity  
@Table(name="StuCourse")  
@Data  
public class StuCourse {  
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")  
 private String StuCourseID;  
   
 /\*\*学生编号\*\*/  
 private String studentID;  
   
 /\*\*课程编号\*\*/  
 private String courseID;  
   
 /\*\*删除标志\*\*/  
 private int deleteState;  
   
 /\*\* 创建者 \*\*/  
 private String createBy;  
   
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
   
 /\*\*创建时间（时间戳）\*\*/  
 private Long createTime;  
   
 /\*\*更新时间\*\*/  
 private Long updateTime;  
   
 public StuCourse()  
 {  
 super();  
 }  
 public StuCourse(StuCourseReq req)  
 {  
 super();  
 this.studentID=req.getStudentID();  
 this.courseID=req.getCourseID();  
 }  
  
}

# Student.java

package com.isl.entity;  
  
  
import java.io.Serializable;  
  
import javax.persistence.EmbeddedId;  
import javax.persistence.Entity;  
import javax.persistence.Table;  
  
import org.springframework.data.annotation.Immutable;  
  
/\*\*  
 \* 学生信息实体类，方便调用  
 \* @author 17678  
 \*  
 \*/  
import lombok.Data;  
  
@Entity  
@Table(name="stu\_courseVIEW")  
@Immutable//只读注解  
@Data  
public class Student implements Serializable{  
  
 /\*\*  
 \*   
 \*/  
 private static final long serialVersionUID = 1L;  
   
 @EmbeddedId  
 private StudentKey studentkey;//联合主键  
 /\*\*课程id\*\*/  
 //private String courseID;  
 /\*\*学生姓名\*\*/  
 private String stuName;//在数据库中的映射字段名为stu\_name  
 /\*\*学生学号\*\*/  
 //private String stuId;//如果是stuID，那么映射为stuid而不是stu\_id  
 /\*\*学生班级\*\*/  
 private String stuClassname;  
 /\*\*学生专业\*\*/  
 private String stuMajorname;  
 /\*\*学生学院\*\*/  
 private String stuAcademyname;  
 /\*\*学生学校\*\*/  
 private String stuSchoolname;  
 /\*\*所选课程名\*\*/  
 private String stuCoursename;  
  
 public Student() {  
 super();  
   
 }  
  
  
   
}

# StudentKey.java

package com.isl.entity;  
  
import java.io.Serializable;  
  
import lombok.Data;  
  
@Data  
public class StudentKey implements Serializable{  
  
 /\*\*  
 \* 联合主键类  
 \*/  
 private static final long serialVersionUID = 1L;  
 private String courseid;  
 private String stuId;  
}

# StuSign.java

package com.isl.entity;  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
  
import com.isl.entity.request.StuSignReq;  
  
import lombok.Data;  
  
  
  
@Entity  
@Table(name="StuSign")  
@Data  
public class StuSign {  
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")  
 private String StuSignID;  
   
 /\*\*学生学号\*\*/  
 private String studentID;  
   
 /\*\*学生微信号\*\*/  
 private String openID;  
 /\*\*签到安排编号\*\*/  
 private String signPlanID;  
   
 /\*\*签到时，二维码的密文\*\*/  
 private String Ciphertext;  
   
 /\*\*学生签到状态 0、未签到1、正常 2、迟到 3、请假 4、缺席 \*\*/  
 private int StusignState;  
   
 /\*\* 创建者 \*\*/  
 private String createBy;  
   
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
   
 /\*\*创建时间（时间戳）\*\*/  
 private Long createTime;  
   
 /\*\*更新时间\*\*/  
 private Long updateTime;  
  
 public StuSign(StuSignReq req)  
 {  
 super();  
 this.openID=req.getOpinID();  
 this.signPlanID=req.getSignPlanID();  
 this.Ciphertext=req.getCiphertext();  
 }  
 public StuSign()  
 {  
 super();  
 }  
}

# User.java

package com.isl.entity;  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import org.hibernate.annotations.GenericGenerator;  
  
import com.isl.entity.request.UserReq;  
  
import lombok.Data;  
  
  
/\*\*  
 \* 用户信息实体类  
 \*   
 \*/  
@Entity  
@Table(name="user")  
@Data  
public class User {  
 @Id  
 @GeneratedValue(generator = "uuid")  
 @GenericGenerator(name = "uuid", strategy = "uuid")  
 private String UserID;  
   
   
 /\*\*学号/工号\*\*/  
 private String RoleID;  
   
 /\*\*密码\*\*/  
 private String password;  
   
 /\*\*用户姓名\*\*/  
 private String RoleName;  
   
 /\*\*用户身份\*\*/  
 private String RoleIdentity;  
   
 /\*\*用户openId\*/  
 private String RoleOpenID;  
   
 /\*\* 创建者 \*\*/  
 private String createBy;  
   
 /\*\* 最近一次更新者 \*\*/  
 private String updateBy;  
   
 /\*\*创建时间（时间戳）\*\*/  
 private Long createTime;  
   
 /\*\*更新时间\*\*/  
 private Long updateTime;  
   
   
 /\*\*删除标志\*\*/  
 private int deleteState;  
   
   
   
 /\*\*班级编号\*\*/  
 private String classID;  
   
 public User()  
 {  
 super();  
 }  
 public User(UserReq req)  
 {  
 this.RoleID=req.getRole\_id();  
 this.RoleName=req.getRole\_name();  
 this.RoleIdentity=req.getRole\_Identity();  
 this.RoleOpenID=req.getRole\_openID();  
 this.classID=req.getClassID();  
 this.password=req.getPassword();  
 }  
 @Override  
 public String toString() {  
 return "User [UserID=" + UserID + ", RoleID=" + RoleID + ", password=" + password + ", RoleName=" + RoleName  
 + ", RoleIdentity=" + RoleIdentity + ", RoleOpenID=" + RoleOpenID + ", createBy=" + createBy  
 + ", updateBy=" + updateBy + ", createTime=" + createTime + ", updateTime=" + updateTime  
 + ", deleteState=" + deleteState + ", classID=" + classID + "]";  
 }  
   
  
}

# AcademyReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.Max;  
import javax.validation.constraints.NotNull;  
import javax.validation.constraints.Size;  
  
import org.hibernate.validator.constraints.Length;  
  
import com.alibaba.excel.annotation.ExcelProperty;  
import com.alibaba.excel.metadata.BaseRowModel;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "academyReq", description = "学院信息")  
@Data  
public class AcademyReq extends BaseRowModel {  
  
 @ExcelProperty(index = 0)  
 @ApiModelProperty(value = "学院的名称", example = "电子与信息工程学院")  
 @NotNull  
 private String academy\_name;  
  
 /\*\* 学校编号 \*\*/  
 @ExcelProperty(index = 1)  
 @ApiModelProperty(value = "学校编号")  
 @NotNull  
 private String schoolID;  
  
}

# ClassReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.NotNull;  
  
import com.alibaba.excel.annotation.ExcelProperty;  
import com.alibaba.excel.metadata.BaseRowModel;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "ClassReq", description = "班级信息")  
@Data  
public class ClassReq extends BaseRowModel {  
 /\*\* 班级名称 \*\*/  
 @ExcelProperty(index = 0)  
 @ApiModelProperty(value = "班级名称", example = "计科174")  
 @NotNull  
 private String className;// 比如计科174  
  
 /\*\* 专业编号 \*\*/  
 @ExcelProperty(index = 1)  
 @ApiModelProperty(value = "专业编号")  
 @NotNull  
 private String majorID;  
}

# CourseReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.NotNull;  
  
import com.alibaba.excel.annotation.ExcelProperty;  
import com.alibaba.excel.metadata.BaseRowModel;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "courseReq", description = "课程信息")  
@Data  
public class CourseReq extends BaseRowModel {  
  
 @ExcelProperty(index = 0)  
 @ApiModelProperty(value = "课程名")  
 @NotNull  
 private String courseName;  
  
 @ExcelProperty(index = 1)  
 @ApiModelProperty(value = "专业名")  
 private String MajorName;  
  
 @ExcelProperty(index = 2)  
 @ApiModelProperty(value = "第一周开始时间", example = "第一周")  
 private String beginTime;  
  
 @ExcelProperty(index = 3)  
 @ApiModelProperty(value = "学时", example = "48")  
 private int hours;  
  
 @ExcelProperty(index = 4)  
 @ApiModelProperty(value = "地点", example = "教二201")  
 private String courseLocation;  
  
 @ExcelProperty(index = 5)  
 @ApiModelProperty(value = "老师编号")  
 private String teacherID;  
  
 @ExcelProperty(index = 6)  
 @ApiModelProperty(value = "管理员编号")  
 private String administratorID;  
  
 @ExcelProperty(index = 7)  
 @ApiModelProperty(value = "课程描述信息ID", example = "String")  
 private String describeID;  
}

# Course\_describeReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.NotNull;  
  
import com.alibaba.excel.annotation.ExcelProperty;  
import com.alibaba.excel.metadata.BaseRowModel;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "Course\_describeReq", description = "课程描述信息")  
@Data  
public class Course\_describeReq extends BaseRowModel {  
  
 /\*  
 \* @ExcelProperty(index = 0)  
 \*   
 \* @ApiModelProperty(value = "课程编号")  
 \*   
 \* @NotNull private String courseID;  
 \*/  
  
 @ExcelProperty(index = 0)  
 @ApiModelProperty(value = "单双周标识", example = "单周/双周/单双周")  
 private String weekSign;  
  
 @ExcelProperty(index = 1)  
 @ApiModelProperty(value = "周几", example = "周一")  
 private String weekTime;  
  
 @ExcelProperty(index = 2)  
 @ApiModelProperty(value = "课程开始时间", example = "10:00")  
 private String courseBeginTime;  
  
 @ExcelProperty(index = 3)  
 @ApiModelProperty(value = "课程结束时间", example = "11:40")  
 private String courseEndTime;  
  
}

# MajorReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.NotNull;  
  
import com.alibaba.excel.annotation.ExcelProperty;  
import com.alibaba.excel.metadata.BaseRowModel;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "MajorReq", description = "专业获取信息")  
@Data  
public class MajorReq extends BaseRowModel {  
  
 /\*\* 专业名称 \*\*/  
 @ApiModelProperty(value = "专业的名称", example = "计算机科学与技术")  
 @NotNull  
 @ExcelProperty(index = 0)  
 private String majorName;  
  
 /\*\* 学院编号 \*\*/  
 @ApiModelProperty(value = "学院编号")  
 @NotNull  
 @ExcelProperty(index = 1)  
 private String academyID;  
  
}

# SchoolReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.NotNull;  
  
import com.alibaba.excel.annotation.ExcelProperty;  
import com.alibaba.excel.metadata.BaseRowModel;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "schoolReq", description = "学校信息")  
@Data  
public class SchoolReq extends BaseRowModel {  
  
 @ApiModelProperty(value = "学校的名称", example = "宁波工程学院")  
 @NotNull  
 @ExcelProperty(index = 0)  
 private String school\_name;  
  
 @ApiModelProperty(value = "学校地址", example = "风华路")  
 @ExcelProperty(index = 1)  
 private String school\_adress;  
  
 @ApiModelProperty(value = "学校邮编", example = "315000")  
 @ExcelProperty(index = 2)  
 private String school\_postcode;  
}

# SignPlanReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.Max;  
import javax.validation.constraints.Min;  
import javax.validation.constraints.NotNull;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value="SignPlanReq",description="签到安排信息")  
@Data  
public class SignPlanReq {  
  
 @ApiModelProperty(value="课程编号")  
 @NotNull  
 private String courseID;  
   
 @ApiModelProperty(value=" 签到持续时间,(min)",example = "5")  
 @NotNull  
 @Min(value = 0)  
 @Max(value = 10)  
 private int sustainTime;//最大10分钟  
   
}

# StuCourseReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.NotNull;  
  
import com.alibaba.excel.annotation.ExcelProperty;  
import com.alibaba.excel.metadata.BaseRowModel;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "StuCourseReq", description = "学生选课信息")  
@Data  
public class StuCourseReq extends BaseRowModel {  
  
 @ApiModelProperty(value = "课程编号")  
 @NotNull  
 @ExcelProperty(index = 0)  
 private String courseID;  
  
 @ApiModelProperty(value = "学生编号")  
 @NotNull  
 @ExcelProperty(index = 1)  
 private String studentID;  
}

# StuSignReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.NotNull;  
  
import com.isl.Valid.IfSignPlan;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "StuSignReq", description = "学生签到信息")  
@Data  
public class StuSignReq {  
 @ApiModelProperty(value = "学生微信号")  
 @NotNull  
 private String opinID;  
  
 @ApiModelProperty(value = "签到安排编号")  
 @NotNull  
 @IfSignPlan()  
 private String signPlanID;  
  
 /\*\* 签到时，二维码的密文 \*\*/  
 @ApiModelProperty(value = "签到二维码密文")  
 @NotNull  
 private String ciphertext;  
  
}

# UserReq.java

package com.isl.entity.request;  
  
import javax.validation.constraints.NotNull;  
import javax.validation.constraints.Size;  
  
import com.alibaba.excel.annotation.ExcelProperty;  
import com.alibaba.excel.metadata.BaseRowModel;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "userReq", description = "用户信息")  
@Data  
public class UserReq extends BaseRowModel {  
  
 @ApiModelProperty(value = "用户学号/工号")  
 @NotNull  
 @ExcelProperty(index = 0)  
 private String role\_id;  
  
 @ApiModelProperty(value = "密码")  
 @ExcelProperty(index = 2)  
 @NotNull  
 @Size(min = 6)  
 private String password;//最少要6位数字  
  
 @ApiModelProperty(value = "用户姓名")  
 @NotNull  
 @ExcelProperty(index = 1)  
 private String role\_name;  
  
 @ApiModelProperty(value = "用户身份", example = "教师/学生/管理员")  
 @ExcelProperty(index = 3)  
 private String role\_Identity;  
  
 @ApiModelProperty(value = "用户绑定id（微信号）")  
 @ExcelProperty(index = 4)  
 private String role\_openID;  
  
 @ApiModelProperty(value = "班级编号")  
 @ExcelProperty(index = 5)  
 private String classID;  
  
 @Override  
 public String toString() {  
 return "UserReq [role\_id=" + role\_id + ", password=" + password + ", role\_name=" + role\_name  
 + ", role\_Identity=" + role\_Identity + ", role\_openID=" + role\_openID + ", classID=" + classID + "]";  
 }  
  
}

# AcademyResp.java

package com.isl.entity.response;  
  
import java.io.Serializable;  
  
import com.isl.entity.Academy;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value="academyResp",description="学院信息")  
@Data  
public class AcademyResp implements Serializable {  
   
 @ApiModelProperty(value="学院信息ID")  
 private String academy\_id;  
   
 @ApiModelProperty(value="学院名称")  
 private String academy\_name;  
   
 public AcademyResp(Academy entity) {  
 super();  
 this.academy\_id = entity.getAcademyID();  
 this.academy\_name = entity.getAcademyName();  
 }  
   
}

# ClassResp.java

package com.isl.entity.response;  
  
import com.isl.entity.Class;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value="ClassResp",description="班级返回信息")  
@Data  
public class ClassResp {  
  
private String classID;//班级编号  
   
 /\*\*班级名称\*\*/  
 @ApiModelProperty(value="班级名称")  
 private String className;//比如计科174  
   
 /\*\*专业编号\*\*/  
 @ApiModelProperty(value="专业编号")  
 private String majorID;  
 public ClassResp(Class entity)  
 {  
 super();  
 this.classID=entity.getClassID();  
 this.className=entity.getClassName();  
 this.majorID=entity.getMajorID();  
 }  
}

# CourseResp.java

package com.isl.entity.response;  
  
import com.isl.entity.Course;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "courseResp", description = "课程信息")  
@Data  
public class CourseResp {  
  
 @ApiModelProperty(value = "课程ID")  
 private String courseID;  
  
 @ApiModelProperty(value = "老师编号")  
 private String teacherID;  
  
 @ApiModelProperty(value = "管理员编号")  
 private String administratorID;  
  
 @ApiModelProperty(value = "专业名")  
 private String majorName;  
  
 @ApiModelProperty(value = "课程名")  
 private String courseName;  
  
 @ApiModelProperty(value = "第一周开始时间")  
 private String beginTime;  
  
 @ApiModelProperty(value = "学时")  
 private int hours;  
  
 @ApiModelProperty(value = "地点")  
 private String courseLocation;  
  
 @ApiModelProperty(value = "课程描述ID")  
 private String describeID;  
  
 public CourseResp(Course entity) {  
 this.courseID = entity.getCourseID();  
 this.teacherID = entity.getTeacherID();  
 this.administratorID = entity.getAdministratorID();  
 this.majorName = entity.getMajorName();  
 this.courseName = entity.getCourseName();  
 this.beginTime = entity.getBeginTime();  
 this.hours = entity.getHours();  
 this.courseLocation = entity.getCourseLocation();  
 this.describeID = entity.getDescribeID();  
 }  
}

# Course\_describeResp.java

package com.isl.entity.response;  
  
  
import com.isl.entity.Course\_describe;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
@ApiModel(value="Course\_describeResp",description="课程描述信息")  
@Data  
public class Course\_describeResp {  
 @ApiModelProperty(value="课程描述编号")  
 private String describeID;  
   
// @ApiModelProperty(value="课程编号")  
// private String courseID;  
   
 @ApiModelProperty(value="单双周标识")  
 private String weekSign;  
   
 @ApiModelProperty(value="周几")  
 private String weekTime;  
   
 @ApiModelProperty(value="课程开始时间")  
 private String courseBeginTime;  
   
 @ApiModelProperty(value="课程结束时间")  
 private String courseEndTime;  
   
 public Course\_describeResp(Course\_describe entity)  
 {  
 this.describeID=entity.getDescribeID();  
// this.courseID=entity.getCourseID();  
 this.weekSign=entity.getWeekSign();  
 this.weekTime=entity.getWeekTime();  
 this.courseBeginTime=entity.getCourseBeginTime();  
 this.courseEndTime=entity.getCourseEndTime();  
 }  
}

# MajorResp.java

package com.isl.entity.response;  
  
import com.isl.entity.Major;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value="MajorResp",description="专业返回信息")  
@Data  
public class MajorResp {  
  
 @ApiModelProperty(value="专业ID")  
 private String majorID;//专业编号  
   
 @ApiModelProperty(value="专业名称")  
 private String majorName;  
   
 @ApiModelProperty(value="学院编号")  
 private String academyID;  
 public MajorResp(Major entity)  
 {  
 super();  
 this.majorID=entity.getMajorID();  
 this.majorName=entity.getMajorName();  
 this.academyID=entity.getAcademyID();  
 }  
}

# SchoolResp.java

package com.isl.entity.response;  
import com.isl.entity.School;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value="schoolResp",description="学校信息")  
@Data  
public class SchoolResp{  
   
 @ApiModelProperty(value="学校信息ID")  
 private String school\_id;  
   
 @ApiModelProperty(value="学校名称")  
 private String school\_name;  
   
   
 @ApiModelProperty(value="学校地址")  
 private String school\_adress;  
   
 @ApiModelProperty(value="学校邮编")  
 private String school\_postcode;  
   
 public SchoolResp(School entity)  
 {  
 super();  
 this.school\_id=entity.getSchoolID();  
 this.school\_name=entity.getSchoolName();  
 this.school\_adress=entity.getSchoolAdress();  
 this.school\_postcode=entity.getSchoolPostcode();  
 }  
}

# SignPlanResp.java

package com.isl.entity.response;  
  
import com.isl.entity.SignPlan;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value="SignPlanResp",description="签到安排信息")  
@Data  
public class SignPlanResp {  
   
 @ApiModelProperty(value="签到安排ID")  
 private String signPlanID;  
   
 @ApiModelProperty(value="课程编号")  
 private String courseID;  
   
 @ApiModelProperty(value="签到开始时间")  
 private String beginTime;  
   
 @ApiModelProperty(value="签到结束时间")  
 private String endTime;  
   
 @ApiModelProperty(value="签到持续时间")  
 private int sustainTime;  
   
   
 @ApiModelProperty(value="签到状态")  
 private String signStatus;  
   
 public SignPlanResp(SignPlan entity)  
 {  
 super();  
 this.signPlanID=entity.getSignPlanID();  
 this.courseID=entity.getCourseID();  
 this.beginTime=entity.getBeginTime();  
 this.endTime=entity.getEndTime();  
 this.signStatus=entity.getSignStatus();  
 this.sustainTime=entity.getSustainTime();  
 }  
}

# StuCourseResp.java

package com.isl.entity.response;  
  
import com.isl.entity.StuCourse;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value="StuCourseResp",description="学生选课信息")  
@Data  
public class StuCourseResp {  
 @ApiModelProperty(value="选课编号")  
 private String StuCourseID;  
   
 @ApiModelProperty(value="学生编号")  
 private String studentID;  
   
 @ApiModelProperty(value="课程编号")  
 private String courseID;  
  
 public StuCourseResp(StuCourse entity)  
 {  
 super();  
 this.StuCourseID=entity.getStuCourseID();  
 this.studentID=entity.getStudentID();  
 this.courseID=entity.getCourseID();  
 }  
}

# StuSignResp.java

package com.isl.entity.response;  
  
import com.isl.entity.StuSign;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "stuSignResp", description = "学生签到信息")  
@Data  
public class StuSignResp {  
  
 @ApiModelProperty(value = "学生学号")  
 private String studentID;  
  
 @ApiModelProperty(value = "签到安排编号")  
 private String signPlanID;  
  
 @ApiModelProperty(value = "签到状态编号")  
 private int StusignState;  
  
 public StuSignResp(StuSign entity) {  
 super();  
 this.signPlanID = entity.getSignPlanID();  
 this.studentID = entity.getStudentID();  
 this.StusignState = entity.getStusignState();  
 }  
}

# StuSignStatusResp.java

package com.isl.entity.response;  
  
import java.io.Serializable;  
  
import com.isl.entity.StuSign;  
import com.isl.entity.User;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value = "stuSignStatusResp", description = "当前课程学生信息")  
@Data  
public class StuSignStatusResp implements Serializable {  
  
 @ApiModelProperty(value = "学生学号")  
 private String studentID;  
  
 @ApiModelProperty(value = "学生姓名")  
 private String RoleName;  
  
 @ApiModelProperty(value = "学生签到状态 1、正常 2、迟到 3、请假 4、缺席")  
 private int StusignState;  
  
 @ApiModelProperty(value = "签到时间（时间戳）")  
 private Long createTime;  
  
 public StuSignStatusResp(User userEntity, StuSign stuSignEntity) {  
 this.setCreateTime(stuSignEntity.getCreateTime());  
 this.setStudentID(stuSignEntity.getStudentID());  
 this.setStusignState(stuSignEntity.getStusignState());  
 this.setRoleName(userEntity.getRoleName());  
 }  
  
}

# UserResp.java

package com.isl.entity.response;  
  
import java.io.Serializable;  
  
import com.isl.entity.User;  
  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
  
@ApiModel(value="userResp",description="用户信息")  
@Data  
public class UserResp implements Serializable{  
 /\*\*  
 \*   
 \*/  
 private static final long serialVersionUID = 1L;  
  
 @ApiModelProperty(value="用户编号")  
 private String UserID;  
  
   
 @ApiModelProperty(value="学号/工号")  
 private String RoleID;  
   
 @ApiModelProperty(value="用户姓名")  
 private String RoleName;  
   
 @ApiModelProperty(value="用户身份")  
 private String RoleIdentity;  
   
 @ApiModelProperty(value="用户openid")  
 private String RoleOpenID;  
   
   
 @ApiModelProperty(value="班级编号")  
 private String classID;  
   
 public UserResp(User entity)  
 {  
 super();  
 this.UserID=entity.getUserID();  
 this.RoleID=entity.getRoleID();  
 this.RoleName=entity.getRoleName();  
 this.RoleIdentity=entity.getRoleIdentity();  
 this.RoleOpenID=entity.getRoleOpenID();  
 this.classID=entity.getClassID();  
 }  
   
}

# ExcelFileReader.java

package com.isl.excel;  
  
import java.io.BufferedInputStream;  
import java.util.List;  
  
import org.springframework.web.multipart.MultipartFile;  
  
import com.alibaba.excel.EasyExcelFactory;  
import com.alibaba.excel.metadata.BaseRowModel;  
import com.alibaba.excel.metadata.Sheet;  
  
public class ExcelFileReader {  
  
 public static <T> List<T> readfile(MultipartFile file,BaseRowModel model) {  
 List<T> data = null;  
 try {  
 BufferedInputStream inputStream = new BufferedInputStream(file.getInputStream());  
  
 data = (List<T>) EasyExcelFactory.read(inputStream, new Sheet(1, 1,model.getClass()));  
   
 for (Object object : data) { System.out.println(object); }  
  
 } catch (Exception e) { // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
 return data;  
 }  
  
}

# CORSFilter.java

package com.isl.filter;  
  
  
import java.io.IOException;  
  
import javax.servlet.Filter;  
import javax.servlet.FilterChain;  
import javax.servlet.FilterConfig;  
import javax.servlet.ServletException;  
import javax.servlet.ServletRequest;  
import javax.servlet.ServletResponse;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import org.springframework.stereotype.Component;  
import lombok.extern.slf4j.Slf4j;  
  
@Component  
@Slf4j  
public class CORSFilter implements Filter{  
  
   
 @Override  
 public void init(FilterConfig filterConfig) throws ServletException {  
   
 }  
  
 @Override  
 public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)  
 throws IOException, ServletException {  
   
   
 HttpServletRequest request = (HttpServletRequest) req;  
 HttpServletResponse response = (HttpServletResponse) resp;  
 //跨域访问  
 response.setHeader("Access-Control-Allow-Origin",request.getHeader("Origin"));//允许任何请求来源  
 response.setHeader("Access-Control-Allow-Headers", "Content-Type,XFILENAME,XFILECATEGORY,XFILESIZE");  
 response.setHeader("Access-Control-Allow-Methods","GET,POST,DELETE,PUT,OPTIONS");  
 response.setHeader("Access-Control-Allow-Credentials","true"); //是否支持cookie跨域  
 //向下一个过滤器转发  
 chain.doFilter(request, response);  
   
 }  
  
 @Override  
 public void destroy() {  
   
 }  
  
}

# SetCharacterEncodingFilter.java

package com.isl.filter;  
  
import java.io.IOException;  
  
import javax.servlet.Filter;  
import javax.servlet.FilterChain;  
import javax.servlet.FilterConfig;  
import javax.servlet.ServletException;  
import javax.servlet.ServletRequest;  
import javax.servlet.ServletResponse;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import org.springframework.stereotype.Component;  
  
import lombok.extern.slf4j.Slf4j;  
  
@Component  
@Slf4j  
public class SetCharacterEncodingFilter implements Filter{  
  
 //编码格式  
 private static String encoding= "UTF-8";  
   
 @Override  
 public void init(FilterConfig filterConfig) throws ServletException {  
 }  
  
 @Override  
 public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)  
 throws IOException, ServletException {  
   
 HttpServletRequest request = (HttpServletRequest) req;  
 HttpServletResponse response = (HttpServletResponse) resp;  
 // 判断当前请求是否单独设置编码格式  
   
 if(request.getCharacterEncoding()==null)  
 {  
   
   
 //获取默认编码格式  
 String encoding = this.encoding;  
 if(encoding!=null)  
 {  
 request.setCharacterEncoding(encoding);  
 response.setContentType("text/plain;charset="+encoding);  
   
 }  
 }else{  
 response.setContentType("text/plain;charset="+request.getCharacterEncoding());  
 }  
   
 //向下一个过滤器转发  
 chain.doFilter(request, response);  
   
 }  
  
 @Override  
 public void destroy() {  
 }  
  
}

# Constants.java

package com.isl.globe;  
  
import java.awt.Color;  
import java.awt.Font;  
  
public class Constants {  
 /\*\*  
 \* 验证码长度  
 \*/  
 public static final Integer IMAGE\_WIDTH=90;  
   
 /\*\*  
 \* 验证码宽度  
 \*/  
 public static final Integer IMAGE\_HEIGHT=30;  
   
 /\*\*  
 \* 验证码字库  
 \*/  
 public static Font[] codeFont={new Font("Times New Roman",Font.PLAIN,20),  
 new Font("Times New Roman",Font.PLAIN,20),new Font("Times New Roman",Font.PLAIN,20),  
 new Font("Times New Roman",Font.PLAIN,20),new Font("Times New Roman",Font.PLAIN,20)};  
 /\*\*  
 \* 验证码每个字符颜色  
 \*/  
 public static Color[] color={Color.BLACK,Color.BLUE,Color.RED,Color.DARK\_GRAY,Color.GREEN};  
   
 /\*\*  
 \* 验证码字库  
 \*/  
 public static final String IMAGE\_CHAR="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";  
   
 /\*\*  
 \* session中的用户ID  
 \*/  
 public static String USER\_ID = "userID";  
 public static String IDENTITY = "identity";  
 public static String NAME = "name";  
 public static String SCHOOL\_ID = "school\_id";  
 public static String ACADEMY\_ID = "academy\_id";  
 public static String MAJOR\_ID = "major\_id";  
 public static String CLASS\_ID = "class\_id";  
 /\*\*  
 \* 验证码  
 \*/  
 public static String CHECK\_NUMBER\_NAME = "identify\_code";  
   
  
 /\*\*  
 \* 保存在ServletContext中的用户列表  
 \*/  
 public static String ONLINE\_USERS = "users";  
   
 /\*\*  
 \* 软件拥有方  
 \*/  
 public static final String SOFTWARE\_OWNER = "宁工信息安全实验室";  
   
 /\*\*  
 \* 默认头像名  
 \*/  
 public static final String DEFAULT\_HEAD\_PICTURE\_NAME = "default.png";  
   
 /\*\*  
 \* 初始化  
 \*/  
 public static final String INIT = "init";  
   
 /\*\*  
 \* 增加  
 \*/  
 public static final String ADD = "add";  
   
 /\*\*  
 \* 删除  
 \*/  
 public static final String DELETE = "delete";  
   
 /\*\*  
 \* 搜索  
 \*/  
 public static final String SEARCH = "search";  
   
 /\*\*  
 \* 更新  
 \*/  
 public static final String UPDATE = "update";  
   
 /\*\*  
 \* 详情  
 \*/  
 public static final String DETAIL = "detail";  
   
 /\*\*  
 \* 重置密码  
 \*/  
 public static final String RESET\_PASSWORD = "resetPwd";  
   
 /\*\*  
 \* 验证请求  
 \*/  
 public static final String IDENTIFY\_CODE = "identify";  
   
 /\*\*  
 \* 用户或者记录状态活跃  
 \*/  
 public static final int STATIC\_ACTIVE=1;  
   
 /\*\*  
 \* 用户或记录状态非法  
 \*/  
 public static final int STATIC\_INVALID=0;  
   
 /\*\*  
 \* 用户或记录不存在  
 \*/  
 public static final int STATIC\_DELETE=-1;  
   
   
 /\*\*  
 \* 不存在任何错误，指向null  
 \*/  
 public static final int NO\_ERROR\_EXIST=-1;  
   
  
 /\*\*  
 \* 该用户名已经被注册  
 \*/  
 public static final int USER\_ACCOUNT\_EXIST = 1;  
 /\*\*  
 \* 该用户名包含非法字符  
 \*/  
 public static final int USER\_ACCOUNT\_CONTAIN\_INVALIDE\_WORD=2;  
   
 /\*\*  
 \* 注册成功  
 \*/  
 public static final int REGISTER\_SUCCESS=3;  
   
 /\*\*  
 \* 由于网络等未知因素，造成注册失败  
 \*/  
 public static final int UNKNOWN\_REGISTER\_ERROR=4;  
   
 /\*\*  
 \* 用户账户不存在  
 \*/  
 public static final int USER\_ACCPUNT\_NOT\_EXIST=5;  
   
 /\*\*  
 \* 登录密码错误  
 \*/  
 public static final int USER\_LOGIN\_PWD\_ERROR=6;  
   
 /\*\*  
 \* 非法访问  
 \*/  
 public static final int INVALID\_REQUEST=7;  
   
 /\*\*  
 \* bean实体创建失败  
 \*/  
 public static final int ENTITY\_INSTANCE\_DEFEAT=8;  
  
 /\*\*  
 \* 当前请求没有对应的资源  
 \*/  
 public static final int INFORMATION\_NOT\_EXIST = 10;  
  
 /\*\*  
 \* 信息更新成功  
 \*/  
 public static final int INFORMATION\_UPDATE\_SECCESS = 11;  
  
 /\*\*  
 \* 由于网络等未知因素导致，信息处理失败，请刷新重试  
 \*/  
 public static final int UNKNOWN\_OPERATION\_ERROR = 12;  
  
 /\*\*  
 \* redis事物被打断，需重新提交  
 \*/  
 public static final int REDIS\_TRANSACTION\_INTERRUPT = 13;  
   
 /\*\*  
 \* 手机号与预留号码不匹配  
 \*/  
 public static final int INVALID\_PHONE\_NUMBER = 14;  
   
 /\*\*  
 \* 验证码错误  
 \*/  
 public static final int IDENTIFY\_CODE\_ERROR=15;  
 /\*\*  
 \* 文件类型有错误  
 \*/  
 public static final int FILE\_TYPE\_ERROR=16;  
 /\*\*  
 \* 文件不存在  
 \*/  
 public static final int FILE\_NOT\_FIND=17;  
 /\*\*  
 \* 不存在任何错误，指向null  
 \*/  
 public static final int NO\_SIGN\_STARTING=18;  
 /\*\*  
 \* 缓存过期时间  
 \*/  
 public static final int CACHE\_SERVICE\_EXPIRE\_TIME = 60;  
 /\*\* --------------------------------------------------------------- \*\*/  
 /\*\* 对象存储失败 \*\*/  
 public static final int REDIS\_ENTITY\_SAVE\_ERROR = 1000;  
 /\*\* 上传信息不完整 \*\*/  
 public static final int INFO\_NOT\_COMPLETE = 1001;  
 /\*\* 对象删除失败 \*\*/  
 public static final int REDIS\_ENTITY\_DELETE\_ERROR = 1002;  
 /\*\* 不可能报出的错，无法得出错误原因 \*\*/  
 public static final int UNKNOWN\_ERROR = 1003;  
 /\*\* 用户类型不存在 \*\*/  
 public static final int USER\_TYPE\_NOT\_EXIST = 1004;  
 /\*\* 邮箱格式不正确 \*\*/  
 public static final int EMAIL\_PATTERN\_NOT\_CORRECT = 1005;  
 /\*\* 手机号格式不正确 \*\*/  
 public static final int PHONE\_PATTERN\_NOT\_CORRECT = 1006;  
 /\*\* 用户名或密码不正确 \*\*/  
 public static final int USERNAME\_OR\_PASSWORD\_NOT\_EXIST = 1007;  
 /\*\* 信息加载失败，请刷新重试 \*\*/  
 public static final int INFORMATION\_LOAD\_DEFEAT=1008;  
 /\*\* 请求信息中包含非法数据 \*\*/  
 public static final int INVALID\_NUMBER\_REQUEST = 1009;  
 /\*\* 请求参数错误 \*\*/  
 public static final int SEARCH\_PARAM\_ERROR = 1010;  
 /\*\* 数据库数据保存出错 \*\*/  
 public static final int DB\_SAVE\_ERROR = 1011;  
 /\*\* 数据库获取数据失败 \*\*/  
 public static final int DB\_GET\_ERROR = 1012;  
 /\*\* 身份错误 \*\*/  
 public static final int IDENTITY\_ERROR = 1013;  
 /\*\* 错误的登录方式 \*\*/  
 public static final int LOGIN\_WAY\_ERROR = 1014;  
 /\*\* openID获取失败 \*\*/  
 public static final int OPENID\_ERROR = 1015;  
}

# MyAdaptableJobFactory.java

package com.isl.quartz;  
  
import org.quartz.spi.TriggerFiredBundle;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.config.AutowireCapableBeanFactory;  
import org.springframework.scheduling.quartz.AdaptableJobFactory;  
import org.springframework.stereotype.Component;  
  
@Component("myAdaptableJobFactory")  
public class MyAdaptableJobFactory extends AdaptableJobFactory {  
 // AutowireCapableBeanFactory 可以将一个对象添加到 SpringIOC 容器中， 并且完成该对象注入  
 @Autowired  
 private AutowireCapableBeanFactory autowireCapableBeanFactory;  
  
 /\*\*\* 该方法需要将实例化的任务对象手动的添加到 springIOC 容器中并且完成对 象的注入 \*/  
 @Override  
 protected Object createJobInstance(TriggerFiredBundle bundle) throws Exception {  
 Object obj = super.createJobInstance(bundle);  
 // 将 obj 对象添加 Spring IOC 容器中，并完成注入  
 this.autowireCapableBeanFactory.autowireBean(obj);  
 return obj;  
 }  
}

# QuartzOne.java

package com.isl.quartz;  
  
import java.util.Date;  
import java.util.List;  
  
import org.quartz.Job;  
import org.quartz.JobExecutionContext;  
import org.quartz.JobExecutionException;  
import org.springframework.beans.factory.annotation.Autowired;  
  
import com.isl.entity.SignPlan;  
import com.isl.service.SignPlanRepository;  
  
public class QuartzOne implements Job {  
 @Autowired  
 public SignPlanRepository signPlanRepository;  
  
// public QuartzOne(SignPlanRepository signPlanRepository,HttpSession session) {  
// this.signPlanRepository=signPlanRepository;  
// this.session=session;  
// }  
 public void doEx() {  
 List<SignPlan> signPlans = this.signPlanRepository.findAll();  
 int count = signPlans.size();  
 for (SignPlan signPlan : signPlans) {  
 String endtime = signPlan.getEndTime();  
 if (Long.valueOf(endtime) < System.currentTimeMillis() / 1000  
 && signPlan.getSignStatus().equals("正在正常签到")) {  
 signPlan.setSignStatus("签到已结束");  
 System.out.println("签到结束");  
 this.signPlanRepository.save(signPlan);  
 System.out.println("修改签到计划状态成功");  
 }  
// if (Long.valueOf(endtime) > System.currentTimeMillis() / 1000)  
// System.out.println("仍在签到期间");  
 }  
// for (SignPlan signPlan : signPlans) {  
// if (signPlan.getSignStatus().equals("签到已结束"))  
// count--;  
// }  
// if (count == 0)  
// System.out.println("所有签到都结束了");  
  
 }  
  
 @Override  
 public void execute(JobExecutionContext context) throws JobExecutionException {  
 //System.out.println("定时正在执行" + new Date());  
 this.doEx();  
  
 }  
  
}

# AcademyRepository.java

package com.isl.service;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import com.isl.entity.Academy;  
  
public interface AcademyRepository extends JpaRepository<Academy,String> {  
  
}

# ClassRepository.java

package com.isl.service;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import com.isl.entity.Class;;  
  
public interface ClassRepository extends JpaRepository<Class, String>{  
   
 @Query(value="select \* from class where classid=? ",nativeQuery=true)  
 public Class findByClassID(String classid);  
  
}

# CourseRepository.java

package com.isl.service;  
  
import java.util.List;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import com.isl.entity.Course;  
  
public interface CourseRepository extends JpaRepository<Course, String>{  
 @Query(value="select \* from course where courseid=?",nativeQuery=true)  
 public Course findByCourseID(String courseid);  
 @Query(value="select \* from course where teacherid=?",nativeQuery = true)  
 public List<Course> findCoursesByTeaId(String tea\_id);  
}

# Course\_describeRepository.java

package com.isl.service;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import com.isl.entity.Course\_describe;  
  
public interface Course\_describeRepository extends JpaRepository<Course\_describe, String>{  
 @Query(value = "select \* from course\_describe where describeid=?",nativeQuery=true)  
 public Course\_describe findByCdID(String describeid);  
}

# MajorRepository.java

package com.isl.service;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import com.isl.entity.Major;  
  
public interface MajorRepository extends JpaRepository<Major, String>{  
  
 @Query(value="select \* from major where majorid=? ",nativeQuery=true)  
 public Major findByMajorID(String major\_id);  
}

# SchoolRepository.java

package com.isl.service;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import com.isl.entity.School;  
  
public interface SchoolRepository extends JpaRepository<School, String>{  
 @Query(value = "select \* from school where schoolid=?",nativeQuery=true)  
 public School findBySchoolID(String schoolid);  
}

# SignPlanRepository.java

package com.isl.service;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.stereotype.Service;  
  
import com.isl.entity.SignPlan;  
@Service  
public interface SignPlanRepository extends JpaRepository<SignPlan, String>{  
 @Query(value = "select \* from sign\_plan where sign\_planid=?",nativeQuery=true)  
 public SignPlan findBySpID(String sp\_id);  
}

# StuCourseRepository.java

package com.isl.service;  
  
import java.util.List;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import com.isl.entity.StuCourse;  
  
public interface StuCourseRepository extends JpaRepository<StuCourse, String> {  
  
 @Query(value = "select \* from stu\_course where stu\_courseid=? ", nativeQuery = true)  
 public StuCourse findByScID(String sc\_id);  
  
 @Query(value = "select \* from stu\_course where studentid=? and courseid=? ", nativeQuery = true)  
 public StuCourse findBySCID(String studentid, String courseid);  
  
 @Query(value = "select studentid from stu\_course where courseid=? ", nativeQuery = true)  
 public List<String> findStuByCourseID(String courseid);  
}

# StudentRepository.java

package com.isl.service;  
  
import java.util.List;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
  
import com.isl.entity.Student;  
import com.isl.entity.StudentKey;  
  
public interface StudentRepository extends JpaRepository<Student, StudentKey>{  
  
 @Query(value="select \* from stu\_courseVIEW where courseid=?",nativeQuery = true)  
 public List<Student> findAllStudents(String course\_id);  
   
}

# StuSignRepository.java

package com.isl.service;  
  
import java.util.List;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import com.isl.entity.StuSign;  
  
public interface StuSignRepository extends JpaRepository<StuSign, String> {  
  
 @Query(value = "select \* from stu\_sign where stu\_signid=?", nativeQuery = true)  
 public StuSign findBySsID(String ss\_id);  
  
 @Query(value = "select \* from stu\_sign where sign\_planid=?", nativeQuery = true)  
 public List<StuSign> findBySignPlanID(String sign\_planid);  
  
 @Query(value = "select \* from stu\_sign where sign\_planid=? and studentid=?", nativeQuery = true)  
 public StuSign findBySSPID(String sign\_planid, String studentid);  
}

# UserRepository.java

package com.isl.service;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import com.isl.entity.User;  
  
public interface UserRepository extends JpaRepository<User, String>{  
   
 @Query(value = "select \* from user where userid=?",nativeQuery=true)  
 public User findByUserID(String userid);  
   
 @Query(value = "select \* from user where roleid=?",nativeQuery=true)  
 public User findByRoleid(String roleid);  
   
 @Query(value = "select \* from user where role\_openid=?",nativeQuery=true)  
 public User findByRoleOpenid(String roleopenid);  
   
 @Query(value="select roleid from user where role\_openid=?",nativeQuery = true)  
 public String findRoleidByOpenid(String openid);  
  
}

# IfSignPlan.java

package com.isl.Valid;  
  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
import javax.validation.Constraint;  
import javax.validation.Payload;  
  
  
  
/\*\*\*  
 \* 校验是否开启签到  
 \* @author 17678  
 \*  
 \*/  
  
 @Constraint(validatedBy = {IfSignPlanValidator.class})  
 @Target({ElementType.ANNOTATION\_TYPE,ElementType.METHOD, ElementType.FIELD})  
 @Retention(RetentionPolicy.RUNTIME)  
 public @interface IfSignPlan{   
 String message() default "该签到安排不存在";  
 String signplanid() default "123" ;   
 Class<?> [] groups() default {} ;   
 Class<? extends Payload>[] payload () default {} ;  
 }

# IfSignPlanValidator.java

package com.isl.Valid;  
  
import javax.validation.ConstraintValidator;  
import javax.validation.ConstraintValidatorContext;  
  
import com.isl.service.SignPlanRepository;  
  
public class IfSignPlanValidator implements   
 ConstraintValidator<IfSignPlan , String>{  
 SignPlanRepository signPlanRepository;  
 public IfSignPlanValidator(SignPlanRepository signPlanRepository) {  
 this.signPlanRepository=signPlanRepository;  
 System.out.println("repository注入");  
 }  
   
 @Override  
 public boolean isValid(String sp\_id, ConstraintValidatorContext context) {  
 System.out.println("验证进行中");  
   
 return this.signPlanRepository.findBySpID(sp\_id)!=null;  
 }  
  
}

# WeChatSession.java

package com.isl.wechat;  
  
import lombok.Data;  
  
@Data  
public class WeChatSession {  
 String openid;  
 String session\_key;  
 String unionid;  
 Number errcode;  
 String errmsg;  
}