源码分`开源版`和`商业版`，这两个版本的功能和授权不同，演示和学习时，先使用`开源版`。

演示工程	代码地址
SpringBlade	https://gitee.com/kuzank/SpringBlade-QuickStart
Saber	https://gitee.com/kuzank/Saber-QuickStart

以下实验例子是在MacBook上进行，文档中的路径会有所不同，望悉知。

- 前端：`Saber`，基于`Vue`的前端架构

  使用技术：Vue、element-ui、AVue、TypeScript

  源码地址：https://gitee.com/smallc/Saber

学习 CSS 布局	**http://zh.learnlayout.com/toc.html**
深入浅出 CSS 布局	http://layout.imweb.io/
element-ui	https://element.eleme.cn/#/zh-CN/component/installation
Vue.js	https://cn.vuejs.org/v2/guide/
Avue	https://avuejs.com/doc/installation
Avue 演示例子列表/表单的Json配置	https://avuejs.com/doc/crud/crud-doc

- 后台：`SpringBlade`，基于`Boot`/`Cloud`的后端架构

  使用技术：Java8、SpringBoot、SpringCloud、Mybatis、Mybatis-Plus、Lombok

  Cloud源码地址：https://gitee.com/smallc/SpringBlade

  Boot源码地址：https://gitee.com/smallc/SpringBlade/tree/2.0-boot

Java8 系列	**https://zhuanlan.zhihu.com/java8**
mybatis	https://mybatis.org/mybatis-3/zh/index.html
MyBatis-Plus	https://mybatis.plus/
纯洁的微笑	http://www.ityouknow.com/
程序员DD	http://blog.didispace.com/

开发工具

- 前端：WebStorm
- 后台：IntelliJ IDEA

1、环境准备

- 公共环境：Git
- 前端环境：Node.js
- 后台环境：Java8、Mysql、Redis、Maven、Lombok【IDE添加此插件】

前端工程安装依赖默认使用国外的镜像，速度比较慢，可通过配置 cnpm 提速：https://www.jianshu.com/p/ad58bbcede05

2、获取源码

```
# 1、创建工作文件夹
mkdir -p /Users/kuzan/Documents/openSource
cd /Users/kuzan/Documents/openSource

# 2、获取前端代码
git clone https://gitee.com/smallc/Saber.git
# 安装前端依赖，通过 npm 或者 cnpm
cd Saber
cnpm install # npm install

# 3、获取boot后台代码
cd /Users/kuzan/Documents/openSource
git clone https://gitee.com/smallc/SpringBlade.git
cd SpringBlade
# 切换为 boot 分支
git checkout 2.0-boot
```

3、初始化数据库

- 1、在 mysql 创建数据库 `blade`
- 2、将 SpringBlade/doc/sql/blade-saber-mysql.sql 导入 `blade` 数据库中
- 3、修改 SpringBlade/src/main/resources/application-dev.yml 配置文件中的数据库账号密码

4、运行系统

- 1、WebStorm 打开 Saber 工程并启动
- 2、IntelliJ IDEA 打开 SpringBlade 工程并启动
- 3、浏览器访问 http://localhost:1888/，输入验证码即可进入系统

5、代码生成例子

bladex 框架可通过设计数据库表单和配置参数，生成相应前后端代码，包括菜单

5.1、设计数据库表单

```sql
CREATE TABLE `blade_student` (
  `id` bigint(64) NOT NULL COMMENT '主键',
  `tenant_id` varchar(12) DEFAULT '000000' COMMENT '租户ID',
  `no` varchar(64) DEFAULT NULL COMMENT '学号',
  `name` varchar(64) DEFAULT NULL COMMENT '姓名',
  `sex` varchar(64) DEFAULT NULL COMMENT '性别',
  `birthday` datetime DEFAULT NULL COMMENT '出生日期',
  `create_user` bigint(64) DEFAULT NULL COMMENT '创建人',
  `create_dept` bigint(64) DEFAULT NULL COMMENT '创建部门',
  `create_time` datetime DEFAULT NULL COMMENT '创建时间',
  `update_user` bigint(64) DEFAULT NULL COMMENT '修改人',
  `update_time` datetime DEFAULT NULL COMMENT '修改时间',
  `status` int(2) DEFAULT NULL COMMENT '状态',
  `is_deleted` int(2) DEFAULT NULL COMMENT '是否已删除',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='学生表';
```

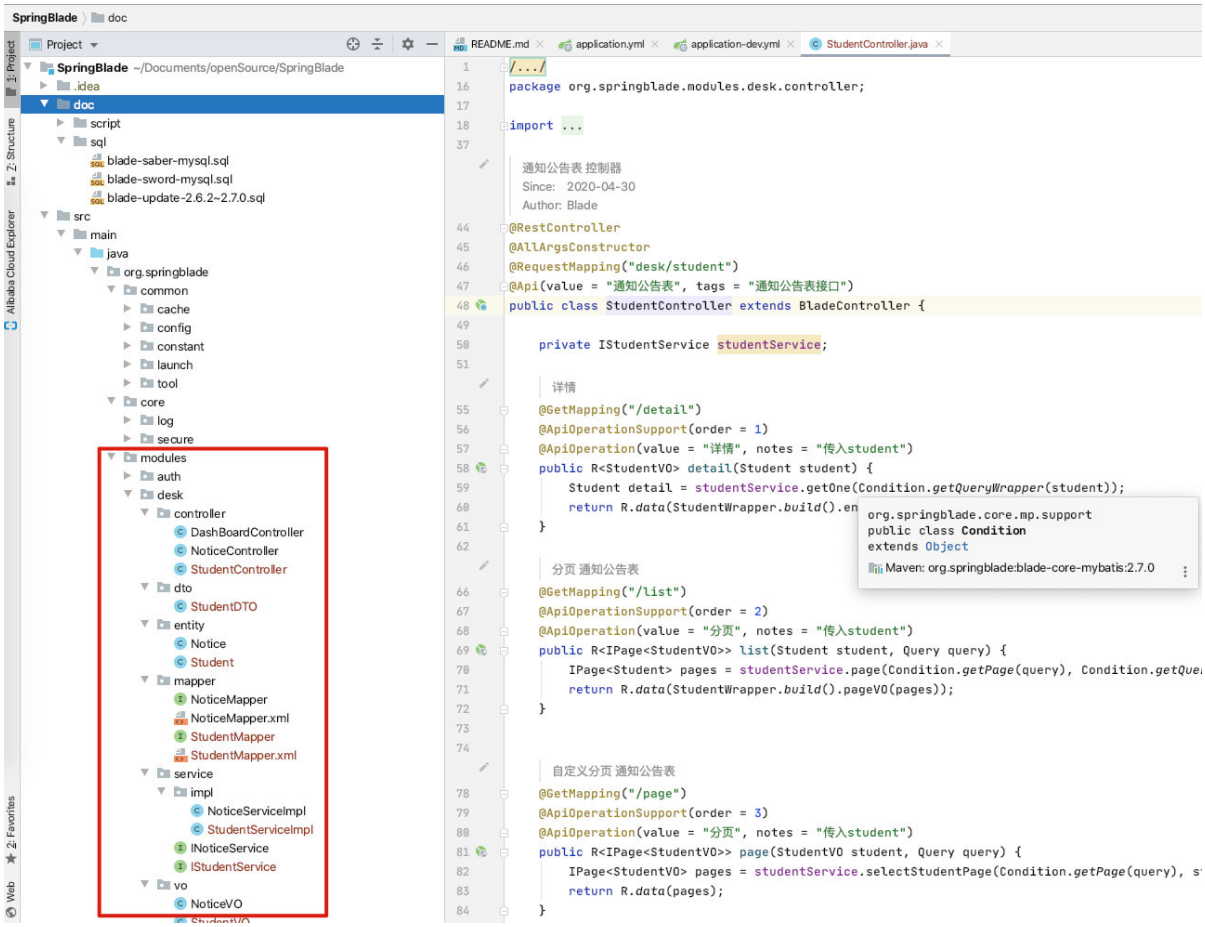
其中，id、tenant_id、create_user、create_dept、create_time、update_user、update_time、status、is_deleted 字段为表单固定的默认字段；

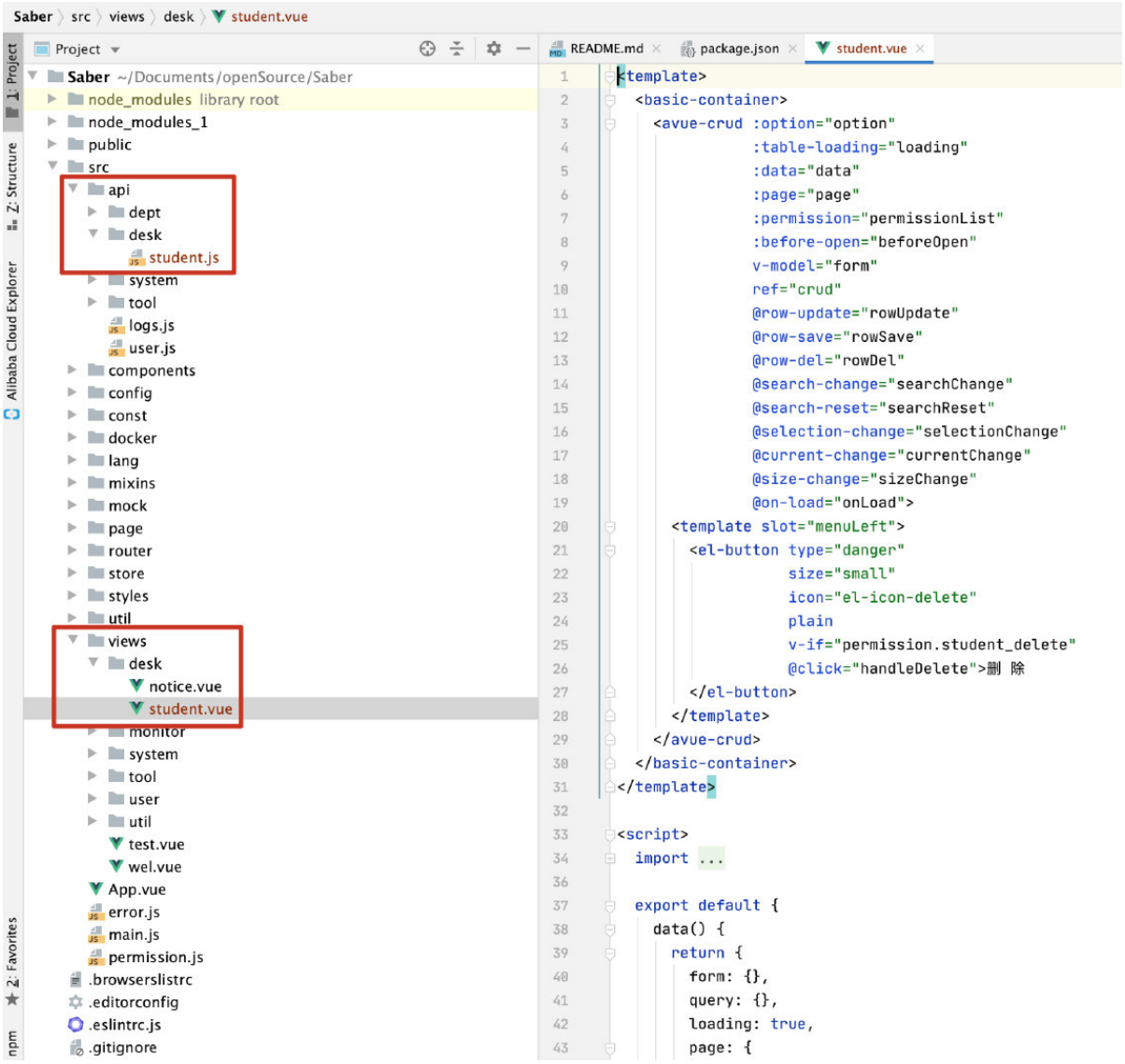
no、name、sex、birthday为表单的业务字段。

5.2、配置参数并生成代码

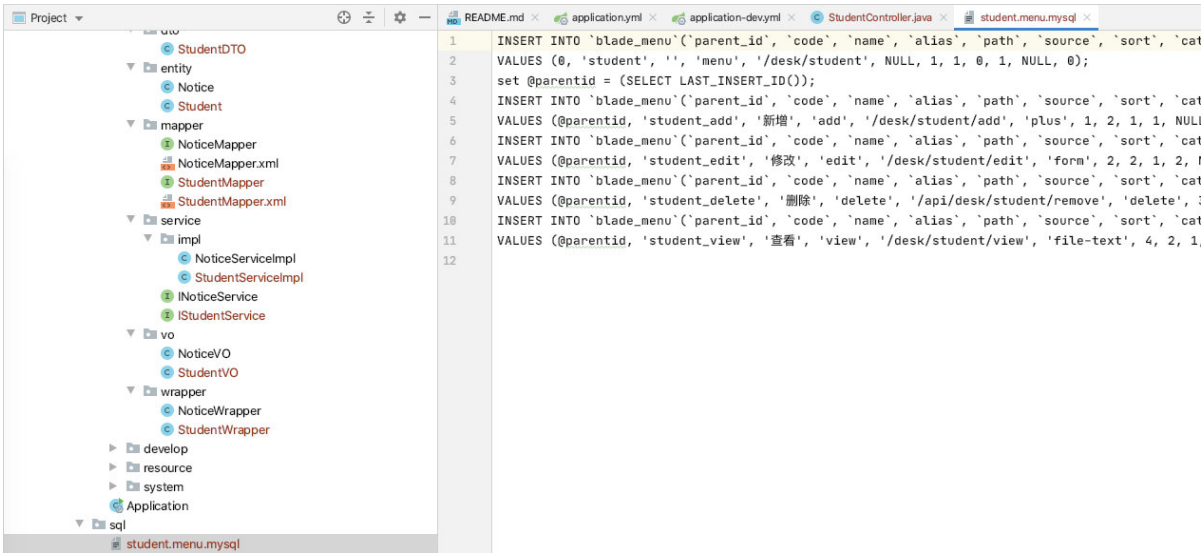
- 浏览器中打开 http://localhost:1888/，登录后进入系统
- 选择菜单 `研发工具` —> `代码生成`，添加代码生成配置，如下图：

- 勾选记录，点击 `代码生成` 按钮
- 查看前端后台工程，可以看到生成的代码，如下图：


SpringBlade
doc
Project
SpringBlade ~/Documents/openSource/SpringBlade
.idea
doc
script
sql
blade-saber-mysql.sql
blade-sword-mysql.sql
blade-update-2.6.2~2.7.0.sql
src
main
java
org.springblade
common
cache
config
constant
launch
tool
core
log
secure
modules
auth
desk
controller
DashBoardController
NoticeController
StudentController
dto
StudentDTO
entity
Notice
Student
mapper
NoticeMapper
NoticeMapper.xml
StudentMapper
StudentMapper.xml
service
impl
NoticeServiceImpl
StudentServiceImpl
INoticeService
IStudentService
vo
NoticeVO
README.md
application.yml
application-dev.yml
StudentController.java
/.../
package org.springblade.modules.desk.controller;
import ...
通知公告表 控制器
Since: 2020-04-30
Author: Blade
@RestController
@AllArgsConstructor
@RequestMapping("desk/student")
@Api(value = "通知公告表", tags = "通知公告表接口")
public class StudentController extends BladeController {
private IStudentService studentService;
详情
@GetMapping("/detail")
@ApiOperationSupport(order = 1)
@ApiOperation(value = "详情", notes = "传入student")
public R<StudentVO> detail(Student student) {
Student detail = studentService.getOne(Condition.getQueryWrapper(student));
return R.data(StudentWrapper.build().en
}
org.springblade.core.mp.support
public class Condition
extends Object
Maven: org.springblade:blade-core-mybatis:2.7.0
分页 通知公告表
@GetMapping("/list")
@ApiOperationSupport(order = 2)
@ApiOperation(value = "分页", notes = "传入student")
public R<IPage<StudentVO>> list(Student student, Query query) {
IPage<Student> pages = studentService.page(Condition.getPage(query), Condition.getQue
return R.data(StudentWrapper.build().pageVO(pages));
}
自定义分页 通知公告表
@GetMapping("/page")
@ApiOperationSupport(order = 3)
@ApiOperation(value = "分页", notes = "传入student")
public R<IPage<StudentVO>> page(StudentVO student, Query query) {
IPage<StudentVO> pages = studentService.selectStudentPage(Condition.getPage(query), s
return R.data(pages);
}



- 将后台生成代码文件中的菜单 sql ，修改一下 sql 语句，然后在数据库 blade 中执行 SpringBlade/src/main/java/sql/student.menu.mysql



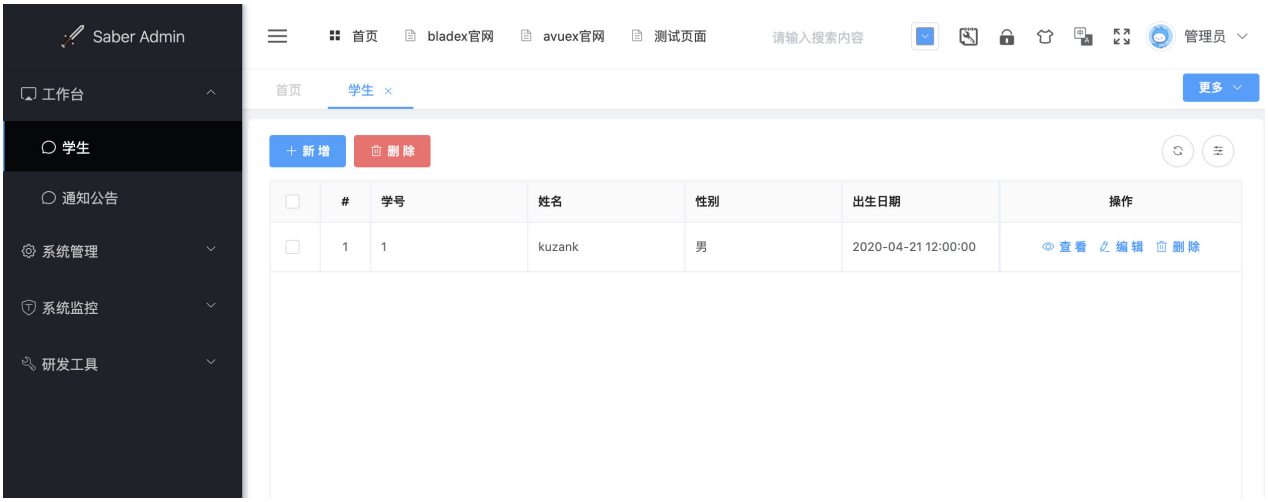
```sql
INSERT INTO `blade_menu`(`id`,`parent_id`, `code`, `name`, `alias`, `path`, `source`, `sort`, `category`, `action`, `is_open`, `remark`, `is_deleted`)
VALUES (1123598812738675202, 0, 'student', '', 'menu', '/desk/student', NULL, 1, 1, 0, 1, NULL, 0);
INSERT INTO `blade_menu`(`id`,`parent_id`, `code`, `name`, `alias`, `path`, `source`, `sort`, `category`, `action`, `is_open`, `remark`, `is_deleted`)
VALUES (1123598812738675203, 1123598812738675202, 'student_add', '新增', 'add', '/desk/student/add', 'plus', 1, 2, 1, 1, NULL, 0);
INSERT INTO `blade_menu`(`id`,`parent_id`, `code`, `name`, `alias`, `path`, `source`, `sort`, `category`, `action`, `is_open`, `remark`, `is_deleted`)
VALUES (1123598812738675204, 1123598812738675202, 'student_edit', '修改', 'edit', '/desk/student/edit', 'form', 2, 2, 1, 2, NULL, 0);
INSERT INTO `blade_menu`(`id`,`parent_id`, `code`, `name`, `alias`, `path`, `source`, `sort`, `category`, `action`, `is_open`, `remark`, `is_deleted`)
VALUES (1123598812738675205, 1123598812738675202, 'student_delete', '删除', 'delete', '/api/blade-desk/student/remove', 'delete', 3, 2, 1, 3, NULL, 0);
INSERT INTO `blade_menu`(`id`,`parent_id`, `code`, `name`, `alias`, `path`, `source`, `sort`, `category`, `action`, `is_open`, `remark`, `is_deleted`)
VALUES (1123598812738675206, 1123598812738675202, 'student_view', '查看', 'view', '/desk/student/view', 'file-text', 4, 2, 1, 2, NULL, 0);
```

5.3 重启后台

代码生成后，需要重启后台系统

5.4 浏览器访问

- 浏览器中打开 http://localhost:1888/，登录后进入系统
- 选择菜单 `系统管理` —> `菜单管理`，修改`学生菜单`的`上级菜单`为`工作台`
- 刷新浏览器查看菜单是否生效
- 退出系统，`一定要` `强制` `刷新` `浏览器页面`，
- 重新登录系统，选择菜单 `工作台` —> `学生`，打开页面，如果无法打开 404，则 `强制` `刷新` `浏览器页面`



5.5 优化前端后台代码

前端列表页面与表单【新增/编辑/查看】页面是有 Json 控制，通过定义/修改 Json 结构，可以修改列表和表单页面，如下所示：

Saber/src/views/desk/student.vue

```
option: {
    height: 'auto',
    calcHeight: 80,
    searchShow: true,
    searchMenuSpan: 6,
    tip: false,
    border: true,
    index: true,
    viewBtn: true,
    selection: true,
    column: [
      {
        label: "学号",
        prop: "no",
        search: true,
        rules: [{
          required: true,
          message: "请输入学号",
          trigger: "blur"
        }]
      },
      {
        label: "姓名",
        prop: "name",
        search: true,
        rules: [{
          required: true,
          message: "请输入姓名",
          trigger: "blur"
```

```
            }]
          },
          {
            label: "性别",
            prop: "sex",
            type: "radio",
            search: true,
            dicData: [
              {
                label: "男",
                value: 'man'
              },
              {
                label: "女",
                value: 'woman'
              },
            ],
            rules: [{
              required: true,
              message: "请输入性别",
              trigger: "blur"
            }]
          },
          {
            label: "出生日期",
            prop: "birthday",
            type: "date",
            format: "yyyy-MM-dd hh:mm:ss",
            valueFormat: "yyyy-MM-dd hh:mm:ss",
            rules: [{
              required: true,
              message: "请输入出生日期",
              trigger: "blur"
            }]
          },
        ]
      },
```

SpringBlade/src/main/java/org/springblade/modules/desk/entity/Student.java

```
package org.springblade.modules.desk.entity;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.fasterxml.jackson.databind.ser.std.ToStringSerializer;
import io.swagger.annotations.ApiModel;
```

```java
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
import lombok.EqualsAndHashCode;
import org.springblade.core.mp.base.BaseEntity;
import org.springframework.format.annotation.DateTimeFormat;

import java.util.Date;

/**
 * 学生表实体类
 *
 * @author Blade
 * @since 2020-04-30
 */
@Data
@TableName("blade_student")
@EqualsAndHashCode(callSuper = true)
@ApiModel(value = "Student对象", description = "学生表")
public class Student extends BaseEntity {

  private static final long serialVersionUID = 1L;

  /**
   * 主键id
   */
  @ApiModelProperty(value = "主键")
  @TableId(value = "id", type = IdType.ASSIGN_ID)
  @JsonSerialize(using = ToStringSerializer.class)
  private Long id;
  /**
   * 学号
   */
  @ApiModelProperty(value = "学号")
  private String no;
  /**
   * 姓名
   */
  @ApiModelProperty(value = "姓名")
  private String name;
  /**
   * 性别
   */
  @ApiModelProperty(value = "性别")
  private String sex;
  /**
   * 出生日期
   */
  @ApiModelProperty(value = "出生日期")
  @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
```

```
59      @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
60      private Date birthday;
61
62  }
```