

瑞吉外卖

Day-----1

搭建数据库

图形界面或则命令行

```
//创建数据库
create database reggie CHARACTER SET utf8mb4
```

导入sql文件命令
`source d:/sql_file/db_reggie.sql`;（路径不要有中文）

搭建maven

maven

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.5</version>
  </parent>
  <groupId>org.hbx.project</groupId>
  <artifactId>reggie</artifactId>
  <version>1.0-SNAPSHOT</version>

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

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

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

```

</dependency>

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

<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-boot-starter</artifactId>
    <version>3.4.2</version>
</dependency>

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.20</version>
</dependency>

<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.76</version>
</dependency>

<dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.6</version>
</dependency>

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>

<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid-spring-boot-starter</artifactId>
    <version>1.1.23</version>
</dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>2.4.5</version>
        </plugin>
    </plugins>
</build>

</project>

```

yaml配置文件

```
server:
  port: 8888
spring:
  application:
    name: reggie_take_out #设置这个项目的名字（默认是项目文件的名字）
  datasource:
    url: jdbc:mysql://localhost:13306/reggie?
    useSSL=false&useUnicode=true&characterEncoding=utf8
    username: root
    password: abc123
    driver-class-name: com.mysql.jdbc.Driver
  mybatis-plus:
    configuration:
      #在映射实体或者属性时，将数据库中表名和字段名中的下划线去掉，按照驼峰命名法映射
      map-underscore-to-camel-case: true
      log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
    global-config:
      db-config:
        id-type: ASSIGN_ID
```

主程序---springbootapplication

导入静态资源-----backed文件和front文件----放在static目录下

或者自己添加静态资源目录

```
@Configuration
public class staticConfig implements WebMvcConfigurer {
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {

        registry.addResourceHandler("/front/**").addResourceLocations("classpath:/front
/");

    }
}
```

后台登陆功能

思路速记

后台登录功能开发

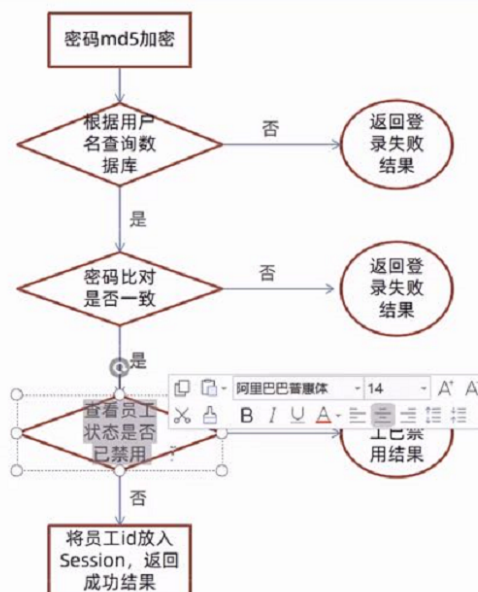


代码开发

4) 在Controller中创建登录方法

处理逻辑如下:

- 1、将页面提交的密码password进行md5加密处理
- 2、根据页面提交的用户名username查询数据库
- 3、如果没有查询到则返回登录失败结果
- 4、密码比对, 如果不一致则返回登录失败结果
- 5、查看员工状态, 如果为已禁用状态, 则返回员工已禁用结果
- 6、登录成功, 将员工id存入Session并返回登录成功结果



controller - service-mapper-查询---返回dao (employee)

前端vue代码的method需要的变量值

规定服务器返回的封装为类型R (code、data、msg、map) ---配合前端

登陆handler---

接受json

保存的session

处理逻辑

(数据库里面是) md5加密 (DigetsUtils.M5D) ----》查用户名-----》查密码

配置过滤器、拦截器---阻止跳过登陆直接访问LoginCheckFilter----- (原生注入的方式)

的好url---筛查是否需要过滤 (/employee/login./dmployee/logout静态资源)

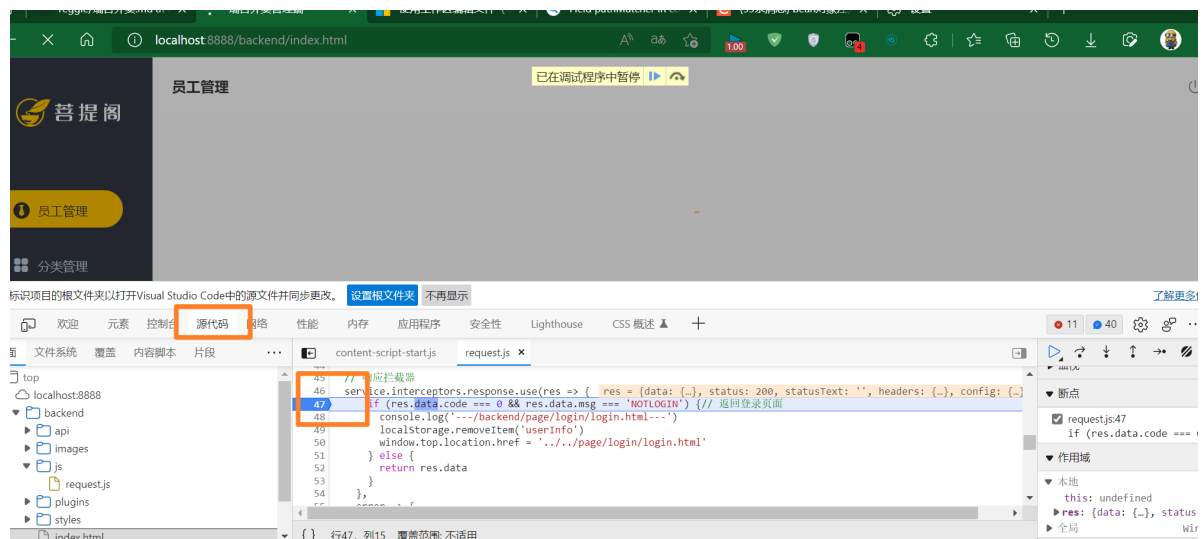
AntPathMatcher-----路径匹配器 解决静态资源 (match方法)

封装check方法检测是否需要放行

获得session对象

没登陆返回---配合JS文件中的响应拦截器-----根据需要的code和msg(后端返回一个输出流 (json的R队形) 就行)

试试调试一下js代码（网页）



```
service.interceptors.response.use(res => {  
  if (res.data.code === 0 && res.data.msg === 'NOTLOGIN') { // 返回登录页面  
    console.log('---/backend/page/login/login.html---')  
    localStorage.removeItem('userInfo')  
    window.top.location.href = '../page/login/login.html'  
  } else {  
    return res.data  
  }  
},
```

backend目录位置导致的问题，因为放在了static文件下，所以还要多退一级目录

```
window.top.location.href = '../../page/login/login.html'  
} else {
```

知识点

mybatisplus

mybatisplus----只需要接口继承父类接口就行

mapper接口继承BaseMapper

service接口继承IService

service实现类继承ServiceImpl<mapper,dao>并且实现service接口

函数介绍

lambdaQuery--eq里面写查询条件

//查询数据库是否右这个用户名

```
LambdaQueryWrapper<Employee> queryWrapper = new LambdaQueryWrapper<>();  
queryWrapper.eq(Employee::getUsername, "admin");  
Employee one = empService.getOne(queryWrapper);
```

注意@Service这些注解是标在实现类上而不是接口上的

html知识

这个方法绑定了登陆按钮

```
methods: {  
  async handleLogin() {  
    this.$refs.loginForm.validate(async (valid) => {  
      if (valid) {  
        this.loading = true  
        let res = await loginApi(this.loginForm)  
        if (String(res.code) === '1') {  
          localStorage.setItem('userInfo', JSON.stringify(res.data))  
          window.location.href= '../..../index.html'  
        } else {  
          this.$message.error(res.msg)  
          this.loading = false  
        }  
      }  
    })  
  }  
}
```

对应的js

```
function loginApi(data) {  
  return $axios({  
    'url': '/employee/login',    control的url  
    'method': 'post',  
    data  
  })  
}
```

发给服务器的是

```
data() {  
  return {  
    loginForm:{  
      username: 'admin',  
      password: '123456'  
    }  
  }  
}
```

是一个json

debug的时候改一下前端的超时时间

Md5工具类--DigestUtils

```
String md5 = DigestUtils.md5DigestAsHex(password.getBytes());
```

后台退出功能

思路速记

右上角展示当前用户

退出按钮的前端代码分析

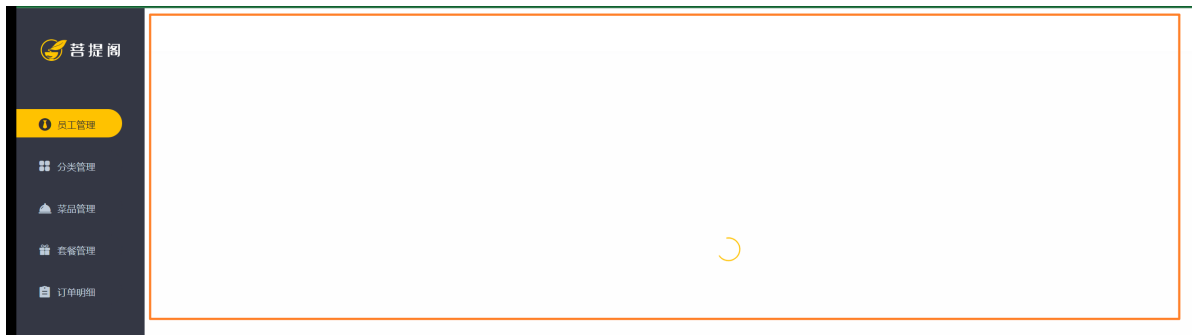
handler

清除缓存

HTML知识点

```
<iframe
  id="cIframe"
  class="c_iframe"
  name="cIframe"
  :src="iframeUrl"
  width="100%"
  height="auto"
  frameborder="0"
  v-show="!loading"
></iframe>
```

相当于留了一片空间用来动态展示src:src="iframeUrl"



利用v-for, v-if, v-else

```
<el-menu-item v-else :index="item.id"
@click="menuHandle(item,false)">
  <i class="iconfont" :class="item.icon"></i>
  <span slot="title">{{item.name}}</span>
</el-menu-item>
```

```
menuList: [
  // {
  //   id: '1',
  //   name: '门店管理',
  //   children: [
  //     {
  //       id: '2',
  //       name: '员工管理',
  //       url: 'page/member/list.html',
  //       icon: 'icon-member'
  //     },
  //   ],
  // },
```



绑定的方法

```
methods: {
  logout() {
    logoutApi().then((res)=>{
      if(res.code === 1){
        localStorage.removeItem('userInfo')
        window.location.href = 'page/login/login.html'
      }
    })
  },
}
```

对应的js

```
function logoutApi(){
  return $axios({
```



```

    'url': '/employee/logout', ----handlerurl
    'method': 'post',
  })
}

```

DAY-----2

新增员工功能

页面发送Ajax请求



```

addEmployee(params).then(res => {
    if (res.code === 1) {    ///-----返回这个就行
        this.$message.success('员工添加成功! ')
        if (!st) {
            this.goBack()
        } else {
            this.ruleForm = {

```

handler接受data

保存数据库

页面没有设置密码，后端给一个默认的初始密码123456，MD5加密，完善employee对象里面的空字段

username唯一约束---所以重名报异常500（使用异常处理器）

@controlleradvice

判断异常信息是否有sql异常信息关键字

截取重复的账号名字

返回错误信息

小知识点

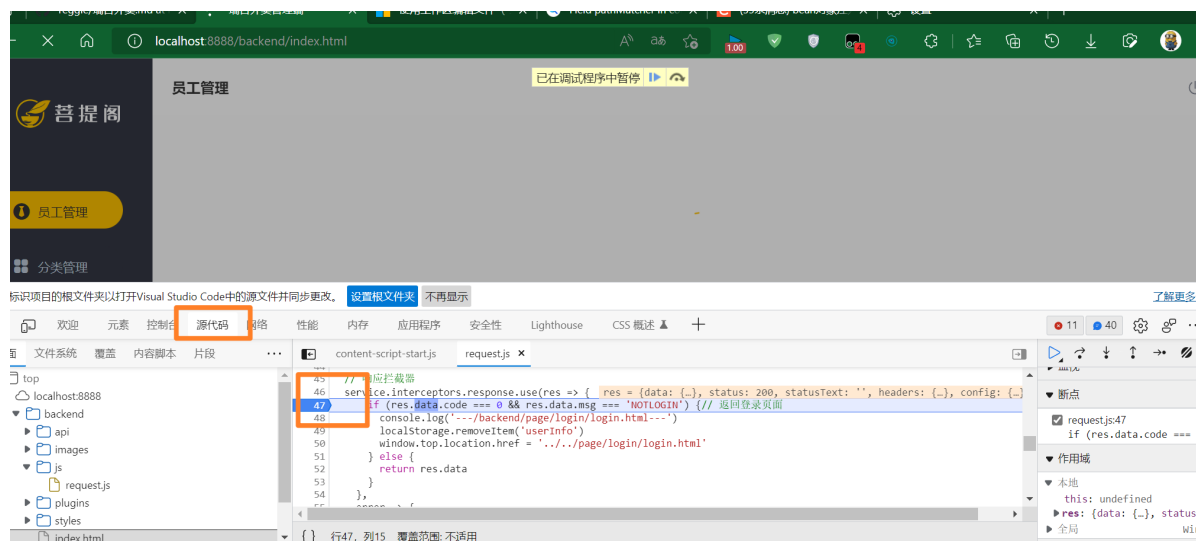
mybatisplus

----save方法-----insert功能

```
empService.save(employee);
```

html

js文件也能进行deug



员工信息分页查询

速记

搜索框

导航栏

ajax请求-----接受参数---查询

getMemverList方法要求后端发的数据

mybatisplus分页插件-----配置分页插件的配置类----MybatisplusInceptor

paginationInnerinterceton

handler--uri--返回类型R《page》----接受参数

构造分页构造器

构造条件构造器

排序条件--orderByDesc----updateTime

执行---page ()

```
//构造分页构造器
Page pageInfo = new Page(page, pageSize); pageInfo: Page@7274 page: 1 pageSize: 10

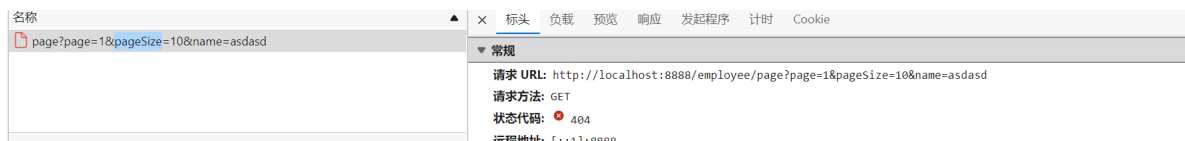
//构造条件构造器
LambdaQueryWrapper<Employee> queryWrapper = new LambdaQueryWrapper();

//添加过滤条件
queryWrapper.like(StringUtils.isEmpty(name), Employee::getName, name);
```

html--修改导航条显示的数量用于测试

employee/page uri

参数-----int page,int pageSize,string name



```
await getMemberList(params).then(res => {
    if (String(res.code) === '1') {
        this.tableData = res.data.records || []
        this.counts = res.data.total
    }-----所以要返回一个page类型的数据
})
js
function getMemberList (params) {
    return $axios({
        url: '/employee/page',
        method: 'get',
        params
    })
}
```

知识点

mybatisplus

分页插件

配置类

```
@Configuration
//@MapperScan("com.hbx.reggie.mapper")-----如果mapper接口上面标注了@Mapper就不要在这里写了
public class MybatisPlusConfig {
    @Bean
    public MybatisPlusInterceptor mybatisPlusInterceptor(){
        MybatisPlusInterceptor interceptor = new MybatisPlusInterceptor();
        interceptor.addInnerInterceptor(new
        PaginationInnerInterceptor(DbType.MYSQL));
        return interceptor;
    }
}
```

```
Page pageInfo = new Page(page, pageSize);
//条件
LambdaQueryWrapper<Employee> employeeLambdaQueryWrapper = new
LambdaQueryWrapper<>();

employeeLambdaQueryWrapper.like(StringUtils.isNotEmpty(name), Employee::getName,
name);

employeeLambdaQueryWrapper.orderByDesc(Employee::getUpdateTime);
empService.page(pageInfo, employeeLambdaQueryWrapper);
```

html

共 3 条 4条/页 < 1 > 前往 1 页

1条/页

2条/页

3条/页

4条/页

```
<el-pagination
  class="pageList"
  :page-sizes="[1, 2, 3, 4]"
  :page-size="pageSize"
```

```
<script>
  new Vue({
    el: '#member-app',
    data() {
      return {
        input: '',
        counts: 0,
        page: 1,
        pageSize: 10,
        tableData : [],
        id : '',
        status : '',
      }
    }
  })

```

启用/禁用员工账号

速记

admin用户才能操作-----为什么---》前端控制的v-if

uri、参数-----》update 数据库 (status、updatetime、updateuser)

返回参数类型---string

maybatisplus----updateById ()

问题-----前端传过来的id有问题----js处理long形数据丢失精度-----》后端将id转换成字符串再发给前端

配置消息转换器---java--》json

jackson

```


    /**
     * @Override
     * protected void extendMessageConverters(List<HttpMessageConverter<?>> converters) {
     *     //创建消息转换器对象
     *     MappingJackson2HttpMessageConverter messageConverter = new MappingJackson2HttpMessageCon
     *     //设置对象转换器，底层使用Jackson将Java对象转为json
     *     messageConverter.setObjectMapper(new JacksonObjectMapper());
     *     //将上面的消息转换器对象追加到mvc框架的转换器集合中
     *     converters.add(index: 0, messageConverter);
     * }
     */

```

▼ 市税

请求 URL: http://localhost:8888/employee

请求方法: PUT

状态代码:  405

响应地址: http://localhost:8888/employee

×

标头

负载

预览

响应

发起程序

计时

Cookie

▼ 请求负载

查看源

▼ {id: 1542793745517342700, status: 0}

id: 1542793745517342700

status: 0

接受参数-----emp对象

```
//状态修改
statusHandle (row) {
  this.id = row.id
  this.status = row.status
  this.$confirm('确认调整该账号的状态?', '提示', {
    'confirmButtonText': '确定',
    'cancelButtonText': '取消',
    'type': 'warning'
  }).then(() => {
    enableOrDisableEmployee({ 'id': this.id, 'status': !this.status ? 1
: 0 }).then(res => {
      console.log('enableOrDisableEmployee',res)
      if (String(res.code) === '1') { ----->返回一个1就行
        this.$message.success('账号状态更改成功! ')
        this.handleQuery()
      }
    })
  })
}
```

知识点

js处理long型数据的会有精度缺失的问题

转换成字符串解决

自定义消息转化器

```
@Override
public void extendMessageConverters(List<HttpMessageConverter<?>>
converters) {
    MappingJackson2HttpMessageConverter mappingJackson2HttpMessageConverter
= new MappingJackson2HttpMessageConverter();
    mappingJackson2HttpMessageConverter.setObjectMapper(new
JacksonObjectMapper());
    converters.add(0,mappingJackson2HttpMessageConverter);
}
```

```
/**
 * 对象映射器:基于jackson将Java对象转为json, 或者将json转为Java对象
 * 将JSON解析为Java对象的过程称为 [从JSON反序列化Java对象]
 * 从Java对象生成JSON的过程称为 [序列化Java对象到JSON]
 */
public class JacksonObjectMapper extends ObjectMapper {

    public static final String DEFAULT_DATE_FORMAT = "yyyy-MM-dd";
    public static final String DEFAULT_DATE_TIME_FORMAT = "yyyy-MM-dd HH:mm:ss";
    public static final String DEFAULT_TIME_FORMAT = "HH:mm:ss";

    public JacksonObjectMapper() {
        super();
        //收到未知属性时不报异常
        this.configure(FAIL_ON_UNKNOWN_PROPERTIES, false);

        //反序列化时，属性不存在的兼容处理

        this.getDeserializationConfig().withoutFeatures(DeserializationFeature.FAIL_ON_
UNKNOWN_PROPERTIES);

        //日期格式化
        SimpleModule simpleModule = new SimpleModule()
            .addDeserializer(LocalDate.class, new
LocalDateDeserialzier(DateTimeFormatter.ofPattern(DEFAULT_DATE_TIME_FORMAT))
        )
            .addDeserializer(LocalDate.class, new
LocalDateDeserialzier(DateTimeFormatter.ofPattern(DEFAULT_DATE_FORMAT)))
            .addDeserializer(LocalTime.class, new
LocalTimeDeserialzier(DateTimeFormatter.ofPattern(DEFAULT_TIME_FORMAT)))
        //long、biginteger等类型转换成json
            .addSerializer(BigInteger.class, ToStringSerializer.instance)
            .addSerializer(Long.class, ToStringSerializer.instance)
            .addSerializer(LocalDate.class, new
LocalDateTimeSerializer(DateTimeFormatter.ofPattern(DEFAULT_DATE_TIME_FORMAT)))
    }
```

```

        .addSerializer(LocalDate.class, new
LocalDateSerializer(DateTimeFormatter.ofPattern(DEFAULT_DATE_FORMAT)))
        .addSerializer(LocalTime.class, new
LocalTimeSerializer(DateTimeFormatter.ofPattern(DEFAULT_TIME_FORMAT)));

    //注册功能模块 例如，可以添加自定义序列化器和反序列化器
    this.registerModule(simpleModule);
}
}

```

mbp

```
empService.updateById(emp);
```

会根据emp对象里面不为空的字符来进行更新

编辑员工信息

速记

编辑按钮绑定的函数-----uri、参数


新增和修改公用add页面

所以有两个handler---一个负责把当前id的信息查询返回给前端显示

▼ 常规

请求 URL: http://localhost:8888/employee/1542793745517342722

请求方法: GET


状态代码:  404

一个负责update (可以和禁用员工的handler共用)

▼ 常规

请求 URL: http://localhost:8888/employee

请求方法: PUT

状态代码:  200

远程地址: [::1]:8888

引用站点策略: strict-origin-when-cross-origin


```
▼ {name: "黄柏轩", phone: "18845678901", sex: "1", idNumber: "123456789012345678", username: "asdasd"}
  idNumber: "123456789012345678"
  name: "黄柏轩"
  phone: "18845678901"
  sex: "1"
  username: "asdasd"
```

```
created() {
    this.id = requestUrlParam('id')
    this.actionType = this.id ? 'edit' : 'add' -----共用
add页面，通过判断是否有id值来区分（修改的有id值）
    if (this.id) {
        this.init()
    }
},
mounted() {
},
methods: {
    async init () {
        queryEmployeeById(this.id).then(res => {
            console.log(res)
            if (String(res.code) === '1') {
                console.log(res.data)
                this.ruleForm = res.data
                this.ruleForm.sex = res.data.sex === '0' ? '女' : '男'
            }
        })
    }
}

//获取url地址上面的参数
function requestUrlParam(argname){
    var url = location.href
    var arrStr = url.substring(url.indexOf("?")+1).split("&")
    for(var i =0;i<arrStr.length;i++)
    {
        var loc = arrStr[i].indexOf(argname+"=")
        if(loc!=-1){
            return arrStr[i].replace(argname+"=", "").replace("?", "")
        }
    }
}
```