技术团队面试系统

项目说明

任务概述

> 目标：搭建一个独立完整的技术团队简历及面试系统。
>
> 项目价值：
>
> 通过逐年推广形成公司技术人才库，沉淀简历资产及过程信息
>
> 便于未来简历再次捞出，挖掘候选人
>
> 便于做年度招聘分析与复盘，优化招聘效果

技术栈

前端：ICE + React

后端：Node.js + Egg + MySQL + React

功能说明

1. 用户管理

1. 账号信息管理
2. 团队成员的权限说明
    a. 超级管理员：管理一切，可增删团队，及设置团队管理员
        i. 皆为软删除（软删除是打标记、硬删除是表数据清除）
    b. 团队管理员：管理成员增删与简历增删（皆为软删除）
        i. 可为团队成员手动重置密码（密码符合一定强度）
    c. 普通成员：查看上传历史，上传简历，其他信息不可见
    d. 面试官：简历增改删、评论、指派（皆为软删除）
        i. 第一个情况，是团队管理者，能参与相关技术面 ii
        ii. 第二个情况，是 HR，能参与面试前后的沟通3
3. 账号登录/登出

2. 团队管理

超级管理员可以做团队增删与修改

1. 可以新增团队，或修改既有团队
2. 新增团队时默认创建团队简历池
    a. 简历池存放待筛选简历
    b. 简历可被废弃（软删除）
3. 设置团队管理员，从管理员池中捞取指定

4. 职位管理

团队管理员 和 超级管理员 可以做 职位岗位增删与修改

5. 可以新增公司的职位
6. 可以新增职位下的岗位要求
7. 将岗位关联到团队下

3. 简历管理

简历新增与指派

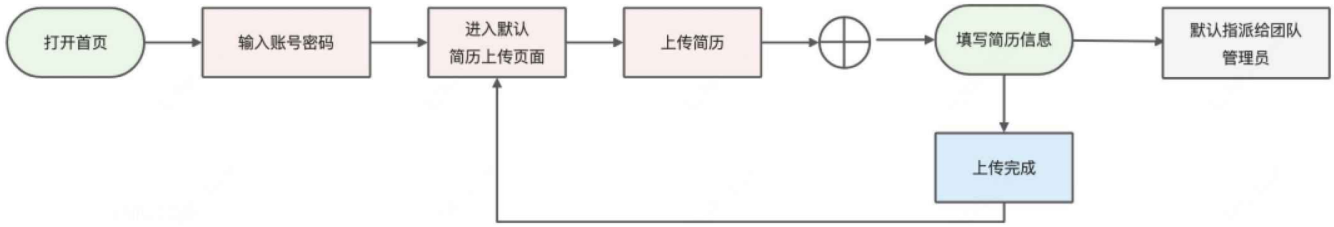
1. 简历新增默认到简历池
    a. 填写简历描述字段
    b. 上传简历附件
    c. 简历上传完毕，自动进入当前登录人上传历史页面
2. 简历进简历池后，默认指派给 团队管理员 筛选
    a. 团队管理员 拿到简历之后，做一下基础筛选（是否有重复简历，是否符合 HC 对应要求...）
    b. 筛选通过，团队管理员 指派给适合的 面试官 ，进入面试流程
    c. 筛选失败， 团队管理员 直接给简历标记为不符合。
3. 简历面试指派
    a. 由 团队管理员 进行 面试官 的指派
    b. 面试官 可修改指派人，将候选人给其他面试官面试
4. 面试管理
5. 面试官面试阶段最多支持到 10 面
    a. 一面、二面...到十面
    b. 每一轮面试需要填写了评价之后，才能进行下一环节指派

c. 如果当时确认当前候选人不适合，面试官可以标记为不合适，直接指派给 HR ，结束当前面试流程

6. 面试评价
   a. 每一轮面试评价中需要包含候选人基础评价和职能评级
   b. 单个环节面试结果可以为不通过，不适合，适合
   c. 单环节面试结果如果为不通过，或者不适合，也可以进入到下一个流程

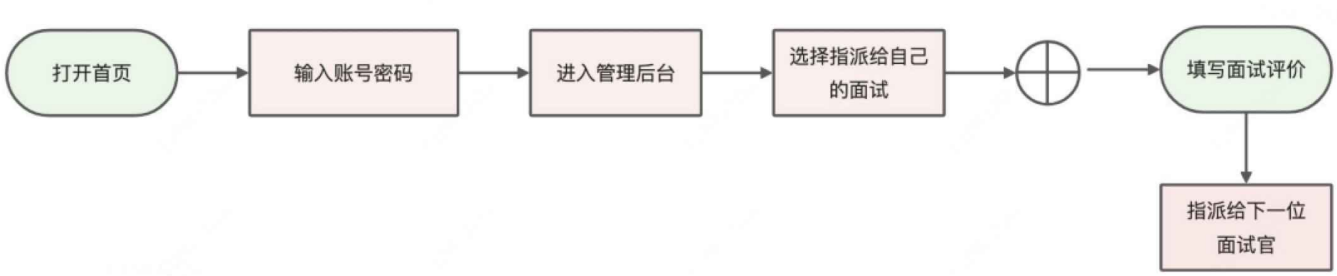
页面访问链路

上传简历流程（所有人）



上传简历环节，简洁版可以只填写字段可以只用填写姓名，手机号，推荐岗位，推荐类型（校招或社招）简历和备注

面试评价流程（所有人）

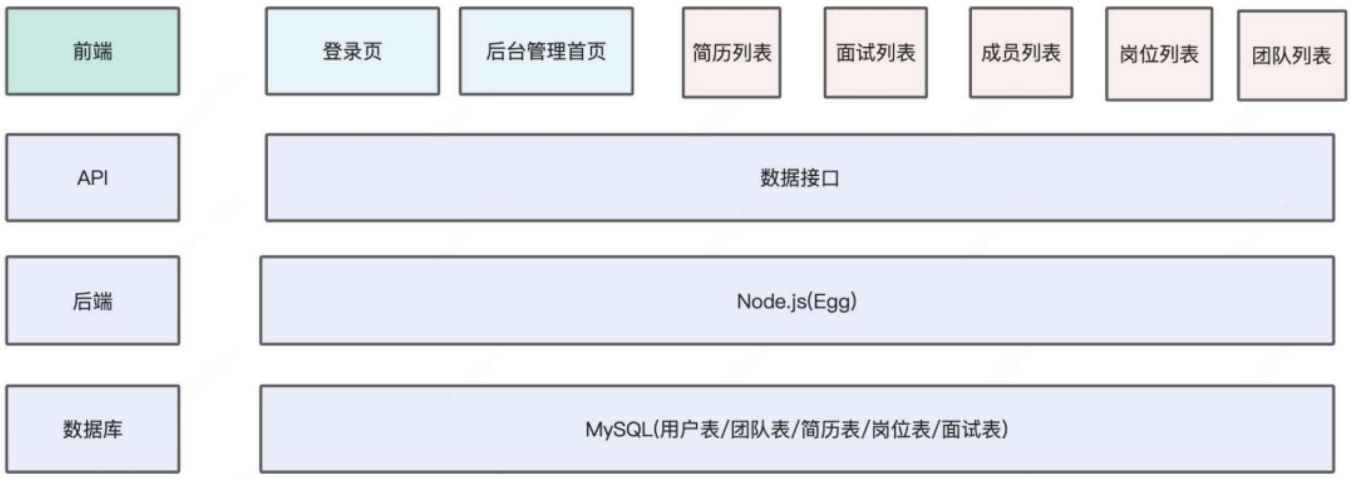


人员新增（团队管理员）

岗位新增（团队管理员）

新增团队及设置管理员（超级管理员）

技术架构



项目里程碑

里程碑 1 - 项目初始化及页面可渲染

可以创建多个分支，参考分支名： ms1/init-project-0302，0302 是建分支日期

- 完成项目环境搭建
- 完成项目的目录文件结构初始化 基于 Egg 的前后端分离项目，能基础跑起来
- 首页/简历详情页的样式完成，可以渲染出登录首页
  - 数据可以用伪造数据展示
  - 注意这个伪数据是 JSON 数据，不是在页面上写固定的数据

验收指标

- 项目目录拆分合理

- 项目的各种配置文件完整
- 后台登录首页可以正常渲染出来

里程碑 2 - 数据建模及成员注册登录

可以创建多个分支，参考分支名：分支名： ms2/schema-and-admin-0302，0302 是建分支日期

- 完成 MySQL 环境搭建与数据库连接
- 完成 用户/简历/面试 等相关数据表结构的 Schema 设计
- 完成用户相关数据表增删改查 API 接口设计
- 完成后台管理页面的开发与渲染
  - 后台首页
  - 新增成员页
  - 成员列表页
  - 新增团队页
  - 团队列表页

验收指标

- 数据建模合理
- 后台页面正常渲染
- 管理员可注册登录
- 团队可被新增、
- 删除、设置管理员
- 成员可以被新增、删除、指派身份、修改密码

里程碑 3 - 简历上传及面试管理

可以创建多个分支，参考分支名：分支名： ms3/articles-sync-0302，0302 是建分支日期

- 完成岗位新增
- 完成简历新增
- 完成面试的筛选功能和指派功能
- 完成面试的评价流程功能
- 完成个人后台中心的关联任务
  - 待筛选的简历 待面试的简历 待评价的简历

验收指标

- 职位可被新增

- 岗位可被新增
- 简历可被上传
- 简历可被指派
- 简历可被评价
- 简历可被删除
- 面试可被指派
- 简历及面试列表、详情可被展示

功能实现注意点

技术简历及面试系统，重点是服务于公司的技术部门，如果希望兼容更多的部门，需要考虑更多的事项，来保证最大的可扩展性，建议是只聚焦在技术部门本身的简历和面试管理（工程师最昂贵，针对工程师的简历管理价值比较容易量化），如下是只聚焦在技术部维度的注意点：

公司的部门及岗位

1. 主要部门维度：技术部、产品部、运营部、人力资源部等
    a. 叫什么无所谓，哪怕产品技术部都可以
        i. 有的工程师还会挂在业务团队中，可以给它一个虚拟的团队名称，如 xx 业务技术部
2. 职位维度：技术、产品、运营、销售、行政、财务、物流、人力资源等
3. 岗位维度：前端实习生、前端工程师、高级前端工程师、资深前端工程师、前端专家、高级前端专家等

简历的几个阶段

1. 简历上传完毕，等待评估
2. 简历过审，状态如下
    a. 等待电话邀约
    b. 等待安排面试
    c. 候选人联系不上
    d. 候选人拒邀

面试的几个阶段

1. 一面、二面、三面

2. ...
3. 面试终止
    a. 面试未通过
    b. 候选人放弃
4. 面试通过
    a. 待下发 Offer
    b. 已下发 Offer
    c. 候选人接纳
    d. 候选人拒接

代码片段

模型能力开发主要是以下几个步骤

模型定义 --> 接口能力实现 --> 路由注册相关能力

业务逻辑控制

可放到 Egg/app/controller 下面

```
const Controller = require('../core/controller')
class IntervieweeController extends Controller {
  async list () {
    const {
      currPage = 1,
      pageSize = 10,
      funcId,
      jobId,
      // 状态，可以传入多个以 , 隔开 status=1,2,3
      status
    } = this.ctx.query
    const res = await this.service.interviewee.pagination({
      currPage,
      pageSize,
      funcId,
      jobId,
      status
    })
    if (res.errMsg) {
      this.fail(res)
```

```
    } else {
      this.success(res)
    }
  }
  /**
评价简历，评价后，面试流程将往后推一步
*/
async comment (ctx) {
const {
// 面试评语
remark,
// 求职者ID
intervieweeId,
// 第几面
step
} = ctx.request.body
// 当前登录用户为面试官
const currentUser = ctx.user
if (!currentUser) {
 this.fail('请登录后再试')
 return
}
// 面试官ID
const userId = currentUser.id
    const viewUserInfo = {
      user_id: userId,
      interviewee_id: intervieweeId,
      status: step,
      comment: remark
    }
    const res = await this.service.interviewee.nextProgress(viewUserInfo)
    this.checkRes(res)
  }
}
module.exports = IntervieweeController
```

MySQL 表字段定义

表结构初始化入库

可放到 Egg/database/migrations 下面

```
'use strict'
const tableName = 'interviewee'
module.exports = {
```

```
  up: async (queryInterface, Sequelize) => {
    const { STRING, INTEGER, DATE, NOW } = Sequelize
    return await queryInterface.createTable(tableName, {
      id: {
        type: INTEGER,
        primaryKey: true,
        autoIncrement: true,
        comment: '主键，自增'
      },
      name: {
        type: STRING,
        allowNull: false,
        comment: '候选人姓名'
      },
      phone: {
        type: STRING,
        allowNull: false,
        unique: 'intervieweePhone',
        is: /^1[3456789]\d{9}$/,
        comment: '候选人电话号码'
      },
      email: {
        type: STRING,
        allowNull: true,
        comment: '候选人电子邮箱',
        isEmail: true
      },
      address: {
        type: STRING,
        allowNull: false,
        comment: '候选人地址'
      },
      education: {
        type: STRING,
        allowNull: false,
        comment: '候选人学历,一般为 高中,大专,本科,研究生,博士,其他'
      },
      type: {
        type: INTEGER(1),
        comment: '招聘类型,0 社招,1校招,默认 0',
        defaultValue: 0
      },
      is_internship: {
        type: INTEGER(1),
        comment: '是否实习 0 试用 1 实习,默认 0',
        defaultValue: 0
      },
```

```
      job_id: {
        type: INTEGER,
        allowNull: true,
        comment: '推荐岗位外键'
      },
      reason: {
        type: STRING,
        allowNull: false,
        comment: '推荐理由'
      },
      resume_path: {
        type: STRING,
        allowNull: false,
        comment: '简历路径'
      },
      note: {
        type: STRING,
        allowNull: true,
        comment: '推荐备注'
      },
      channel: {
        type: INTEGER,
        allowNull: false,
        comment: '招聘渠道 0 外部 1 内部,默认 0',
        defaultValue: 1
      },
      status: {
        type: STRING,
        allowNull: false,
        comment: '面试状态,一面,二面,三面,四面,五面,六面,发放 offer,正式入职'
      },
      state: {
        type: INTEGER(1),
        defaultValue: 1,
        comment: '数据是否有效,0 无效,1 有效,默认 1'
      },
      is_success: {
        type: INTEGER(1),
        defaultValue: 0,
        comment: '是否成功 0 不成功 1 成功'
      },
      viewer_id:{
        type: INTEGER,
        allowNull: true,
        comment: '面试官外键'
      },
      recommender_id: {
```

```
        type: INTEGER,
        allowNull: true,
        comment: '内推人外键'
      },
      created_at: {
        type: DATE,
        allowNull: true,
        defaultValue: NOW,
        comment: '创建时间'
      },
      updated_at: {
        type: DATE,
        allowNull: true,
        defaultValue: NOW,
        comment: '更新时间'
      },
      deleted_at: {
        type: DATE,
        allowNull: true,
        comment: '删除时间'
      },
    }, {
      paranoid: true,
      deletedAt: true,
      underscored: true,
      freezeTableName: true,
      tableName: tableName,
      version: true,
      comment: '候选人表'
    }).then(() =>{
      queryInterface.addIndex(tableName, {
        name: 'name_index',
        fields: ['name'],
      })
      queryInterface.addIndex(tableName, {
        name: 'phone_index',
        fields: ['phone'],
      })
      queryInterface.addIndex(tableName, {
        name: 'status_index',
        fields: ['status'],
      })
    })
  },
  down: async (queryInterface, Sequelize) => {
    return await queryInterface.dropTable(tableName)
  }
```

```
145 }
```

表结构模型定义

放到 Egg/app/model 下面

```
1  const tableName = 'interviewee'
2
3  module.exports = app => {
4    const { STRING, INTEGER } = app.Sequelize
5    const Interviewee = app.model.define(tableName, {
6      id: {
7        type: INTEGER,
8        primaryKey: true,
9        autoIncrement: true,
10       comment: '主键，自增'
11     },
12     name: {
13       type: STRING,
14       allowNull: false,
15       comment: '候选人姓名'
16     },
17     phone: {
18       type: STRING,
19       allowNull: false,
20       unique: 'interviewee Phone',
21       is: /^1[3456789]\d{9}$/,
22       comment: '候选人电话号码'
23     },
24     email: {
25       type: STRING,
26       allowNull: true,
27       comment: '候选人电子邮箱',
28       isEmail: true
29     },
30     address: {
31       type: STRING,
32       allowNull: false,
33       comment: '候选人地址'
34     },
35     education: {
36       type: STRING,
37       allowNull: false,
38       comment: '候选人学历,一般为 高中,大专,本科,研究生,博士,其他'
39     },
```

```
    type: {
      type: INTEGER(1),
      comment: '招聘类型,0 社招,1校招,默认 0',
      defaultValue: 0
    },
    is_internship: {
      type: INTEGER(1),
      comment: '是否实习 0 试用 1 实习,默认 0',
      defaultValue: 0
    },
    job_id: {
      type: INTEGER,
      allowNull: true,
      comment: '推荐岗位外键'
    },
    reason: {
      type: STRING,
      allowNull: false,
      comment: '推荐理由'
    },
    resume_path: {
      type: STRING,
      allowNull: false,
      comment: '简历路径'
    },
    note: {
      type: STRING,
      allowNull: true,
      comment: '推荐备注'
    },
    channel: {
      type: INTEGER,
      allowNull: false,
      comment: '招聘渠道 0 外部 1 内部,默认 0',
      defaultValue: 1
    },
    status: {
      type: STRING,
      allowNull: false,
      comment: '面试状态,一面,二面,三面,四面,五面,六面,发放 offer,正式入职'
    },
    state: {
      type: INTEGER(1),
      defaultValue: 1,
      comment: '数据是否有效,0 无效,1 有效,默认 1'
    },
    is_success: {
```

```
      type: INTEGER(1),
      defaultValue: 0,
      comment: '是否成功 0 不成功 1 成功'
    },
    viewer_id: {
      type: INTEGER,
      allowNull: true,
      comment: '面试官外键'
    },
    recommender_id: {
      type: INTEGER,
      allowNull: false,
      comment: '内推人外键'
    }
  }, {
    tableName: tableName,
    comment: '候选人表',
    hooks: {
    }
  })
  Interviewee.associate = () => {
    Interviewee.belongsTo(app.model.Job, {
      as: 'job',
      foreignKey: 'job_id',
      constraints: false
    })
    Interviewee.belongsTo(app.model.Users, {
      as: 'viewr',
      foreignKey: 'viewer_id',
      constraints: false
    })
    Interviewee.belongsTo(app.model.Users, {
      as: 'recommender',
      foreignKey: 'recommender_id',
      constraints: false
    })
  }
  Interviewee.sync({ alter: true })
  return Interviewee
}
```

接口能力实现

controller 能力扩展说明

可放到 Egg/app/core/controller.js 里

```
const Controller = require('egg').Controller

class BaseController extends Controller {
  validate () {
    const { ctx } = this
    ctx.validate(this.getResRule(ctx.url))
  }
  // 校验用户权限
  checkUseAuth (type) {
    return this.ctx.checkUseAuth(type)
  }
  // set success data
  success (data) {
    this.ctx.success(data)
  }
  successMsg (msg) {
    this.ctx.successMsg(msg)
  }
  // set failed msg
  fail (error) {
    this.ctx.fail(error)
  }
  failMsg (msg) {
    this.ctx.failMsg(msg)
  }
  // get params
  getParams (key) {
    return this.ctx.getParams(key)
  }
  // get request client ip
  get ip () {
    return this.ctx.ip
  }
  // get cookies
  get cookies () {
    return this.ctx.cookies
  }
  // get session
  get session () {
    return this.ctx.session
  }
  validatePk () {
    this.ctx.validate({
      id: { type: 'int', convertType: 'number', required: true }
    }, this.ctx.params)
  }
```

```
  validatePageParam (type = 'query') {
    const checkPath = type === 'body' ? 'body' : 'query'
    this.ctx.validate(this.ctx.rule.getPageRequest,
this.ctx.request[checkPath])
  }
  formatOptions (obj) {
    const badParamList = [undefined, '', null]
    for (const name in obj) {
      badParamList.includes(obj[name]) && delete obj[name]
    }
    return obj
  }
  async find (id) {
    const table = await this.service[this.serviceName].findByPk(id, {}, true)
    if (!table) {
      this.failMsg(
${this.modelName} 表内无标识为 ${id} 的记录
)
    }
    return table
  }
  async destroy (needTip = true) {
    const { ctx } = this
    const id = ctx.params.id
    const table = await this.find(id)
    if (table) {
      await table.destroy()
      needTip && this.successMsg(
删除${id}成功
)
      return true
    }
  }
  // 封装统一的调用检查函数，可以在查询、创建和更新等 Service 中复用
  checkSuccess (result) {
    const DEAULT_REQ_STATUS_ATTR = this.ctx.getAttr('DEAULT_REQ_STATUS_ATTR')
    const DEAULT_REQ_MSG_ATTR = this.ctx.getAttr('DEAULT_REQ_MSG_ATTR')
    if (!result[DEAULT_REQ_STATUS_ATTR]) {
      this.ctx.failMsg(result[DEAULT_REQ_MSG_ATTR])
    } else {
      this.ctx.successMsg(result[DEAULT_REQ_MSG_ATTR])
    }
  }
  checkFindData (table, errMsg = '记录未找到') {
    if (!table) {
      this.failMsg(errMsg)
    }
```

```
    this.success(table)
  }
}
module.exports = BaseController
```

controller 能力消费

可放到 Egg/app/controller里

```
const BaseController = require('../core/controller')

const currServiceName = 'sysDicItems'
/**
@Controller
*/
class SysDicItemsController extends BaseController {
  get modelName () {
    return 'SysDicItems'
  }
  get serviceName () {
    return currServiceName
  }
  /**
@Summary 获取指定分组的所有字典项
@Description 获取指定分组的所有字典项
@Router get /sysDicItems/group/{groupName}
@Request path string groupName 分组名
@Response 200 sys_dicItems 获取指定分组的所有字典项
*/
async group () {
const { ctx } = this
const { params } = ctx
const { groupName } = params
    ctx.validate(ctx.rule.getSysDicGroupRequest)
    const res = await this.service[currServiceName].group(groupName)
    res && this.success(res)
  }
  /**
@Summary 获取所有字典项，不分页
@Description 获取所有字典项
@Router get /sysDicItems
@Response 200 sys_dicItems 获取所有字典项成功
*/
async index () {
 this.success(await this.service[currServiceName].list())
```

```
}
  /**
@Summary 分页获取字典项
@Description 分页获取字典项
@Router post /sysDicItems/page/
@Request query integer pageNo 页面数
@Request query integer pageSize 页面条数
@Response 200 sys_dicItems 获取所有字典项成功
*/
async page () {
this.validatePageParam()
const { ctx } = this
const {
  // 默认第一页
  pageNo = 1,
  // 默认每页10条记录
  pageSize = 10,
  ...restParams
} = ctx.request.body
const where = this.formatOptions({
  ...restParams
})
ctx.validate(ctx.rule.updateSysDicRequest)
const config = { pageNo, pageSize, where }
const res = await this.service[currServiceName].pagination(config)
if (res.count) {
  this.success(res)
} else {
  this.failMsg(
未找到第 ${pageNo} 页的字典项信息
)
}
}
  /**
@Summary 新增字典项
@Description 新增字典项信息
@Router post /sysDicItems
@Request body createSysDicRequest *body
@Response 200 defaultBaseResponse 新增字典项成功
*/
async create () {
 const { ctx } = this
 ctx.validate(ctx.rule.createSysDicRequest)
 const { name, value, note, groupName, sortNum, enable } = ctx.request.body
 const res = await this.service[currServiceName].create({ name, value, note,
groupName, sortNum, enable })
 this.success({ id: res.id })
```

```
}
  /**
@Summary 获取所有字典项
@Description 根据 id 获取字典项信息
@Router get /sysDicItems/{id}
@Request path integer id 标识
@Response 200 sys_dicItems 获取指定字典项信息成功
*/
async show () {
const { ctx } = this
this.validatePk()
const res = await this.find(ctx.params.id)
res && this.success(res)
}
  /**
@Summary 查找字典项
@Description 查找字典项信息
@Router post /sysDicItems/find
@Request body findSysDicRequest *body
@Response 200 sys_dicItems 查找字典项成功
*/
async findByParam () {
const { ctx } = this
const { name, value, note, groupName, sortNum, enable } = ctx.request.body
    // 参数校验
    ctx.validate(ctx.rule.findSysDicRequest, ctx.request.body)
    const where = this.formatOptions({ name, value, note, groupName, sortNum,
enable })
    // 查找字典项数据
    const res = await ctx.service[currServiceName].list({ where })
    return res ? this.success(res) : this.failMsg(
已存在字典项 ${name}
)
  }
  /**
@Summary 修改指定字典项
@Description 根据 id 修改字典项信息
@Router put /sysDicItems/{id}
@Request path integer id 标识
@Request body createSysDicRequest *body
@Response 200 defaultBaseResponse 修改指定字典项成功
*/
async update () {
const { ctx } = this
this.validatePk()
const id = ctx.params.id
const SysDicItems = await this.find(id)
```

```
129 if (SysDicItems) {
130  ctx.validate(ctx.rule.createSysDicRequest)
131  const { name, value, note, groupName, sortNum, enable } = ctx.request.body
132  await this.service[currServiceName].modelUpdateData({
133    model: SysDicItems, data: { name, value, note, groupName, sortNum, enable }
134  })
135  this.success({ id })
136 }
137 }
138   /**
139 @Summary 删除指定字典项
140 @Description 根据 id 删除字典项信息
141 @Router delete /sysDicItems/{id}
142 @Request path integer id 标识，可以用逗号隔开
143 @Response 200 defaultResponse 删除指定字典项成功
144 */
145 async destroy () {
146 const id = this.ctx.params.id
147 const state = await this.service[currServiceName].destroy({ where: { id:
    id.split(',') } })
148 state && this.successMsg(
149 删除字典项${this.ctx.params.id}成功
150 )
151 }
152 }
153 module.exports = SysDicItemsController
```

获取数据服务

services 能力扩展说明

可放到 Egg/app/core/services.js 里

```
1  const Service = require('egg').Service
2  const dayjs = require('dayjs')
3  const humps = require('humps')
4
5  const _symbolJudge = (key, convert) => {
6    if (typeof key === 'symbol') {
7      return key
8    } else {
9      return convert(key)
10   }
11 }
12 function toInt (str) {
```

```
  if (typeof str === 'number') return str
  if (!str) return str
  return parseInt(str, 10) || 0
}

class BaseService extends Service {
  get DEAULT_STATUS_LIST () {
    return this.ctx.getAttr('DEAULT_STATUS_LIST')
  }
  get DEAULT_REQ_STATUS_ATTR () {
    return this.ctx.getAttr('DEAULT_REQ_STATUS_ATTR')
  }
  get DEAULT_REQ_MSG_ATTR () {
    return this.ctx.getAttr('DEAULT_REQ_MSG_ATTR')
  }
  get DEAULT_REQ_DATA_ATTR () {
    return this.ctx.getAttr('DEAULT_REQ_DATA_ATTR')
  }
  get model () {
    return this.ctx.model
  }
  get _user () {
    return this.ctx.user
  }
  // 格式化数据库返回的列表数据，将之从下划线转为驼峰
  formatDBDataToCamelize (data) {
    if (data && data.length > 0) {
      return data.map((item) => humps.camelizeKeys(item.dataValues))
    } else if (data && data.dataValues) {
      return humps.camelizeKeys(data.dataValues)
    }
    return data
  }
  // 格式化下划线为驼峰
  formatDataToCamelize (data) {
    if (data && data.length > 0) {
      return data.map((item) => humps.camelizeKeys(item))
    } else if (data) {
      return humps.camelizeKeys(data)
    }
    return data
  }
  // 格式化驼峰参数为下划线参数，便于查询
  foramtToUnderscoredParams (data) {
    if (data && data.length > 0) {
      return data.map((item) => humps.decamelizeKeys(item, _symbolJudge))
    } else if (data && data) {
```

```
60        return humps.decamelizeKeys(data, _symbolJudge)
61      }
62      return data
63    }
64    formatDate (time) {
65      return dayjs(time).format('YYYY-MM-DD HH:mm:ss')
66    }
67    formatDay (time) {
68      return dayjs(time).format('YYYY-MM-DD')
69    }
70    formatTimestampsField (table, callBackFn = (item) => item) {
71      if (!table) {
72        return table
73      }
74      const CREATE_ATTR = 'createdAt'
75      const UPDATE_ATTR = 'updatedAt'
76      const DELETE_ATTR = 'deletedAt'
77      // const VERSION_ATTR = 'version'
78      table[CREATE_ATTR] && (table[CREATE_ATTR] =
    this.formatDate(table[CREATE_ATTR]))
79      table[UPDATE_ATTR] && (table[UPDATE_ATTR] =
    this.formatDate(table[UPDATE_ATTR]))
80      DELETE_ATTR in table && (table[DELETE_ATTR] = undefined)
81      // this.ctx.checkUseAuth(0) && VERSION_ATTR in table && delete
    table[VERSION_ATTR]
82      return callBackFn(table)
83    }
84    dealWithTableField (table, callBackFn) {
85      return table.dataValues &&
    this.formatTimestampsField(this.formatDataToCamelize(table.dataValues),
    callBackFn)
86    }
87    async dealDBFn (callback, errHandler) {
88      try {
89        return await callback()
90      } catch (e) {
91        errHandler && await errHandler()
92        this.ctx.throw(e.errors ? e.errors[0]?.message : e.message)
93      }
94    }
95    async updateData ({ modelName, data, configOption }) {
96      const _t = this
97      return await _t.dealDBFn(async () => {
98        const entity = _t.foramtToUnderscoredParams(data)
99        return _t.dealWithTableField(await _t.model[modelName].update(entity,
    configOption))
100     })
```

```
101   }
102   async modelUpdateData ({ model, data, configOption }) {
103     const _t = this
104     return await _t.dealDBFn(async () => {
105       const entity = _t.foramtToUnderscoredParams(data)
106       return _t.dealWithTableField(await model.update(entity, configOption))
107     })
108   }
109   async createData ({ modelName, data, configOption, callBackFn }) {
110     const _t = this
111     return await _t.dealDBFn(async () => {
112       const entity = _t.foramtToUnderscoredParams(data)
113       const res = await _t.model[modelName].create(entity, configOption)
114       return res ? _t.dealWithTableField(res, callBackFn) : res
115     })
116   }
117   async findAll ({ modelName = '', configOption = {}, callBackFn }) {
118     const _t = this
119     return await _t.dealDBFn(async () => {
120       const res = await _t.model[modelName].findAll(configOption)
121       return res ? res.map(item => _t.dealWithTableField(item, callBackFn)) :
    res
122     })
123   }
124   async findByPk ({ modelName = '', id = '', configOption = {}, needModal =
    false, callBackFn }) {
125     const _t = this
126     return await _t.dealDBFn(async () => {
127       const res = await _t.model[modelName].findByPk(toInt(id), configOption)
128       res && (res.dataValues = _t.dealWithTableField(res, callBackFn))
129       return needModal ? res : res.dataValues
130     })
131   }
132   async findOne ({ modelName = '', configOption = {}, needModal = false,
    callBackFn }) {
133     const _t = this
134     return await _t.dealDBFn(async () => {
135       const res = await _t.model[modelName].findOne(configOption)
136       res && (res.dataValues = _t.dealWithTableField(res, callBackFn))
137       return needModal ? res : res && res.dataValues
138     })
139   }
140   async findOrCreate ({ modelName = '', configOption = {}, callBackFn }) {
141     const _t = this
142     return await _t.dealDBFn(async () => {
143       const res = await _t.model[modelName].findOrCreate(configOption)
144       return res ? _t.dealWithTableField(res, callBackFn) : res
```

```
145     })
146   }
147   async destroyData ({ modelName = '', configOption = {} }) {
148     const _t = this
149     return await _t.dealDBFn(async () => {
150       await _t.model[modelName].destroy(configOption)
151       return true
152     })
153   }
154   async pagination ({ modelName = '', configOption = {}, needTranUnder = true,
    callBackFn }) {
155     const _t = this
156     const { attributes, include, order } = configOption
157     let { pageNo, pageSize, where } = configOption
158     pageNo = toInt(pageNo)
159     pageSize = toInt(pageSize)
160     where && needTranUnder && (where = _t.foramtToUnderscoredParams(where))
161     return await _t.dealDBFn(async () => {
162       const options = {
163         attributes,
164         include,
165         where,
166         order: order || [
167           ['updated_at', 'DESC']
168         ],
169         offset: (pageNo - 1) * pageSize,
170         limit: toInt(pageSize)
171       }
172       const res = await _t.model[modelName].findAndCountAll(options)
173       const data = res.count > 0 ? res.rows.map(item =>
    _t.dealWithTableField(item, callBackFn)) : []
174       return { data, count: res.count }
175     })
176   }
177   async findAndCountAll ({ modelName = '', configOption = {}, callBackFn }) {
178     const _t = this
179     return await _t.dealDBFn(async () => {
180       const res = await _t.model[modelName].findAndCountAll(configOption)
181       return res ? res.map(item => _t.dealWithTableField(item, callBackFn)) :
    res
182     })
183   }
184   // set success data
185   success (msg, data) {
186     return this.ctx.successMsgResponse(msg, data)
187   }
188   // set failed msg
```

```
189   fail (error) {
190     return this.ctx.errorMsgResponse(error)
191   }
192 }
193 module.exports = BaseService
```

services 能力消费举例

可放到 Egg/app/service/ 里

```
1  const Service = require('egg').BaseService
2  const fs = require('fs')
3  const path = require('path')
4
5  const currModelName = 'Interviewee'
6  /**
7  候选人表  Service
8  */
9  class Interviewee extends Service {
10   get configOption () {
11     return {
12       include: [
13         {
14           as: 'recommender',
15           model: this.model.Users,
16           attributes: ['name']
17         },
18         {
19           as: 'job',
20           model: this.model.Job,
21           attributes: ['name']
22         }
23       ],
24       where: this.getAuthWhere()
25     }
26   }
27   get callBackFn () {
28     return (item) => {
29       item.jobName = item.job && item.job.dataValues.name
30       item.job = undefined
31       // item.viewerName = item.viewr && item.viewr.dataValues.name
32       // item.viewr = undefined
33       item.recommenderName = item.recommender &&
   item.recommender.dataValues.name
34       item.recommender = undefined
```

```
35        return item
36      }
37    }
38    getAuthWhere () {
39      return this.ctx.checkUseAuth(0, false) ? {} : { recommender_id:
    this._user.id }
40    }
41    async list () {
42      return await super.findAll({ modelName: currModelName, configOption:
    this.configOption, callBackFn: this.callBackFn })
43    }
44    async findByPk (id, configOption, needModal) {
45      configOption = { ...this.configOption, ...configOption }
46      configOption.where.id = id
47      this.ctx.checkUseAuth(1) && delete configOption.where.recommender_id
48      return await super.findOne({ modelName: currModelName, configOption,
    needModal, callBackFn: this.callBackFn })
49    }
50    async create (intervieweeInfo, file) {
51      return await this.dealDBFn(async () => {
52        const filename =
53  ${intervieweeInfo.name}-${intervieweeInfo.phone}
54        const uploadRes = await this.uploadPdf(this.ctx.origin, file, filename)
55        intervieweeInfo.resumePath = uploadRes.resumePath
56        // 新增候选人记录
57        const currUser = this._user || {}
58        intervieweeInfo.recommenderId = currUser.id || 1
59        return await this.createData({ modelName: currModelName, data:
    intervieweeInfo })
60      })
61    }
62    async update (data, configOption, needJoin = true) {
63      needJoin && (configOption = { ...this.configOption, ...configOption })
64      let callBackFn = ''
65      needJoin && (callBackFn = this.callBackFn)
66      return await this.updateData({ modelName: currModelName, data,
    configOption, callBackFn })
67    }
68    async modelUpdate (model, data, configOption = {}) {
69      let transaction
70      return await this.dealDBFn(async () => {
71        transaction = await this.model.transaction()
72        configOption.transaction = transaction
73        const res = await super.modelUpdateData({ model, data, configOption })
74        // 维护推荐记录
75        recommendRecordInfo.id = res.recommenderId
76        recommendRecordInfo.intervieweeName = res.name
```

```
77        recommendRecordInfo.intervieweeTel = res.phone
78        recommendRecordInfo.intervieweeType = res.type
79        await this.service.recommendRecord.update(recommendRecordInfo, {
transaction }, false)
80        await transaction.commit()
81        return res
82      }, () => transaction.rollback())
83    }
84    async destroy () {
85      let transaction
86      return await this.dealDBFn(async () => {
87        transaction = await this.model.transaction()
88        const id = this.ctx.params.id
89        let recommenderId = ''
90        this.ctx.checkUseAuth(0) && (recommenderId = this._user.id)
91        const configOption = {
92          where: { id, recommender_id: recommenderId },
93          transaction
94        }
95        // 查找候选人记录
96        let res = await this.exist(configOption.where)
97        if (!res) {
98          return this.ctx.failMsg(
99  ${currModelName} 表内无标识为 ${id} 的记录
100 )
101       }
102       // 删除候选人记录
103       res = await this.destroyData({ modelName: currModelName, configOption })
104       // 删除推荐记录
105       const recommendRecordWhere = {
106         interviewee_id: id,
107         recommender_user_id: recommenderId
108       }
109       await this.service.recommendRecord.destroy({ transaction, where:
recommendRecordWhere })
110       await transaction.commit()
111       return res
112     }, () => transaction.rollback())
113   }
114   /**
115 上传简历
116 @returns {string} res 上传路径
117 */
118 async uploadPdf (origin, stream, filename) {
119 const resumeFolderPath =
120 public/${this.formatDay()}
121 const saveFolder =
```

```
app/${resumeFolderPath}
const resumePath =
${resumeFolderPath}/${filename}.pdf
// 如果文件夹不存在，则直接创建
if (!fs.existsSync(saveFolder)) {
 fs.mkdirSync(saveFolder)
}
const writerStream = fs.createWriteStream(path.join(this.config.baseDir,
${saveFolder}/${filename}.pdf
))
await stream.pipe(writerStream)
return { resumePath, uploadPath:
${origin}${resumePath}
 }
}
  async pagination ({ where, ...otherConfig }) {
    const _t = this
    const { Op } = this.app.Sequelize
    const { createdAt } = where
    where = _t.foramtToUnderscoredParams({ ...where, ..._t.configOption.where
})
    createdAt && (delete where.created_at) && createdAt.length &&
(where.created_at = {
      [Op.lt]: new Date(createdAt[1]),
      [Op.gt]: new Date(createdAt[0])
    })
    // [Op.like]: '%hat',
    return await super.pagination({ modelName: currModelName, configOption: {
...otherConfig, ..._t.configOption, where }, needTranUnder: false, callBackFn:
this.callBackFn })
  }
  async findAllByCondtion (configOption, needJoin = true) {
    needJoin && (configOption = { ...this.configOption, ...configOption })
    let callBackFn = ''
    needJoin && (callBackFn = this.callBackFn)
    return await this.findAll({ modelName: currModelName, configOption,
callBackFn })
  }
  /**
判断简历是否存在
@return true 表示已存在，false 不存在
*/
async exist (where) {
// 判断是否已存在
const configOption = {
 attributes: ['id'],
 where: this.foramtToUnderscoredParams(where)
```

```
}
return !!await super.findOne({ modelName: currModelName, configOption })
}
}
module.exports = Interviewee
```

路由文件配置

路由约束说明

可放到 Egg/app/router.js 配置

```
const { readdirSync } = require('fs')
const { extname, resolve } = require('path')

module.exports = app => {
  const checkFileType = (file) => extname(file) === '.js'
  const BASE_ROUTER = resolve(__dirname, './routers')
  readdirSync(BASE_ROUTER, 'utf-8')
    .filter(checkFileType)
    .map(file => {
      console.log(
[${app.config.projectName}] insert router ${file}
)
      require(resolve(BASE_ROUTER, file))(app)
    })
}
```

路由能力注册举例说明

可放到 Egg/app/routers/

```
module.exports = app => {
  const { router, controller } = app
  router.post('/interviewee/upload', controller.interviewee.upload)
  router.post('/interviewee/page/:param?', controller.interviewee.page)
  router.resources('Interviewee', '/interviewee', controller.interviewee)
  // 获取所有职能(id, name)列表，不分页
  // GET /interviewee             app.controllers.interviewee.index
  // GET /interviewee/:id           app.controllers.interviewee.show
  // PUT /interviewee/:id           app.controllers.interviewee.update
  // DELETE /interviewee/:id      app.controllers.interviewee.destroy
}
```