kkb-honor-admin

玩家信息管理前端实现

主要实现玩家信息查询、新增、更新和删除。

玩家信息查询

分页查询玩家信息并能够按照账户名称过滤数据。

新增玩家信息路由，创建@/router/modules/heros.ts

```
import { RouteConfig } from 'vue-router'
import Layout from '@/layout/index.vue'

export const playerRoutes: RouteConfig = {
  path: '/players',
  component: Layout,
  redirect: '/players/list',
  meta: {
    title: 'playerMgt', // i18n信息需要额外处理
    icon: 'people' // 图标选取
  },
  children: [
    {
      path: 'list',
      component: () => import(/* webpackChunkName: "player-list" */
'@/views/player/list.vue'),
      name: 'PlayerList',
      meta: {
        title: 'playerList', // i18n信息需要额外处理
        icon: 'list' // 图标选取
      }
    }
  ]
}
```

> 在router/index.ts中引入

新增玩家列表页面，创建@/views/player/list.vue

```
<template>
  <div>
    player list
  </div>
</template>

<script lang="ts">
  import { Component, Vue } from 'vue-property-decorator'
  @Component
  export default class extends Vue {}
</script>
```

> 测试，菜单i18n需要额外处理

国际化处理 - i18n

我们项目中利用vue-i18n实现国际化。

以中文定义为例说明，修改lang/zh.ts：

```
export default {
  // 在route里面添加heroMgt对应导航内容
  route: {
    heroMgt: '英雄管理',
    playerList: '玩家列表',
  }
```

它能生效是因为导航菜单项中使用route作为前缀获取相应内容，

layout/components/Sidebar/SidebarItem.vue

```
{{ $t('route.' + theOnlyOneChild.meta.title) }}
```

> 诸如Breadcrumb、TagsView中也有类似使用

使用图标

icon模块列出了内置的所有图标，如果没有中意的，可以扩展图标库：

1. 从iconfont复制图标内容并保存至`src/icons/svg`中

2. 执行`yarn svg`重新生成组件

获取玩家列表

编写getPlayers接口，创建api/players.ts

```
import request from '@/utils/request'

export const getPlayers = (params: any) =>
  request({
    url: '/players',
    method: 'get',
    params
  })
```

查询玩家数据，player/list.vue

```
import { getPlayers } from '@/api/players'
import { Player } from '@/api/types'

export default class extends Vue {
  // 玩家数据
  private list: Player[] = [];
  // 加载状态
  private listLoading = true;

  created() {
    this.getList()
  }
  // 获取列表
  private async getList() {
    this.listLoading = true
    const { data } = await getPlayers()
    this.list = data.items
    this.total = data.total
    this.listLoading = false
  }
}
```

数据展示，player/list.vue

```
<div class="app-container">
```

```
  <el-table
      v-loading="listLoading"
      :data="list"
      border
      fit
      highlight-current-row
      style="width: 100%"
    >
      <el-table-column
        align="center"
        label="ID"
      >
        <template v-slot="{row}">
          <span>{{ row.id }}</span>
        </template>
      </el-table-column>

      <el-table-column
        align="center"
        label="登录账户"
      >
        <template v-slot="{row}">
          <span>{{ row.acountname }}</span>
        </template>
      </el-table-column>
  </el-table>
</div>
```

分页，player/list.vue

```
@Component({
  name: 'PlayerList',
  components: {
    Pagination
  }
})
export default class extends Vue {
  private total = 0; // 总条数

  // 查询参数
  private listQuery = {
    page: 1,
    limit: 20,
  };

  private async getList() {
    // 传入查询参数
```

```
    const { data } = await getPlayers(this.listQuery)
  }
}
```

```
<pagination
  v-show="total>0"
  :total="total"
  :page.sync="listQuery.page"
  :limit.sync="listQuery.limit"
  @pagination="getList"
/>
```

条件查询，player/list.vue

```
export default class extends Vue {
  // 增加accountname做查询条件
  private listQuery = {
    acountname: undefined,
  }
  // 触发过滤
  private handleFilter() {
    this.listQuery.page = 1
    this.getList()
  }
}
```

```
<div class="filter-container">
  <el-input
            v-model="listQuery.acountname"
            :placeholder="$t('player.accountname')"
            style="width: 200px;"
            class="filter-item"
            @keyup.enter.native="handleFilter"
            />
  <el-button
             v-waves
             type="primary"
             icon="el-icon-search"
             @click="handleFilter"
             >
    {{ $t('player.search') }}
  </el-button>
</div>
```

新增玩家

常见交互方式有两种：弹窗、独立页面

添加路由配置，router/modules/players.ts

```
{
  path: 'create',
  component: () => import('@/views/player/create.vue'),
  name: 'CreatePlayer',
  meta: {
     title: 'createPlayer',
     icon: 'edit'
  }
},
```

创建views/player/create.vue

```
<template>
  <div>
    player create
  </div>
</template>
```

导航

```
<el-button
   type="primary"
   icon="el-icon-edit"
   @click="handleCreate"
>
  {{ $t('player.add') }}
</el-button>
```

```
private handleCreate() {
  this.$router.push('/players/create')
}
```

编写创建玩家表单，views/player/create.vue

```
<template>
  <!--创建一个p;ayer-detail组件，更新时可复用-->
  <player-detail :is-edit="false" />
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator'
import PlayerDetail from './components/PlayerDetail.vue'

@Component({
  name: 'CreatePlayer',
  components: {
    PlayerDetail
  }
})
export default class extends Vue {}
</script>
```

创建views/player/components/player-detail.vue

```
<template>
  <div>
    <el-form
      ref="playerForm"
      :model="playerForm"
      :rules="rules"
    >
      <el-form-item prop="accountname" label="账户名">
        <el-input v-model="playerForm.acountname"/>
      </el-form-item>

      <el-form-item prop="nickname" label="昵称">
        <el-input v-model="playerForm.nickname"/>
      </el-form-item>

      <el-form-item>
        <el-button type="primary" @click="submitForm">
            提交
        </el-button>
      </el-form-item>
    </el-form>
  </div>
</template>

<script lang="ts">
import { Component, Prop, Vue } from 'vue-property-decorator'
import {
```

```
  getPlayer,
  updatePlayer,
  createPlayer,
  defaultPlayerData
} from '@/api/players'
import { Form } from 'element-ui'

@Component({
  name: 'PlayerDetail'
})
export default class extends Vue {
  @Prop({ default: false })
  private isEdit!: boolean;

  // 初始化数据，默认均为空
  private playerForm = Object.assign({}, defaultPlayerData);
  private loading = false;

  created() {
    // 如果是更新数据则获取对应玩家信息
    if (this.isEdit) {
      const id = this.$route.params && this.$route.params.id
      this.fetchData(parseInt(id))
    }
  }

  private async fetchData(id: number) {
    try {
      const { data } = await getPlayer(id, {})
      this.playerForm = data.player
    } catch (err) {
      console.error(err)
    }
  }

  private submitForm() {
    // 加载状态
    this.loading = true

    try {
      // 提交操作
      if (this.isEdit) {
        await updatePlayer(this.playerForm.id, this.playerForm)
      } else {
        await createPlayer(this.playerForm)
      }

      // 操作成功
      this.$notify({
```

```
        title: '操作成功',
        message: '新增玩家数据成功',
        type: 'success',
        duration: 2000
      })

      // 还原加载状态
      this.loading = false
    } catch (error) {
      console.error(error)
    }
  }
}
</script>
```

头像上传

```
<el-form-item prop="avatar" label="用户头像">
  <el-upload
      class="avatar-uploader"
      action="https://jsonplaceholder.typicode.com/posts/"
      :show-file-list="false"
      :on-success="handleAvatarSuccess"
      :before-upload="beforeAvatarUpload">
     <img v-if="imageUrl" :src="imageUrl" class="avatar">
     <i v-else class="el-icon-plus avatar-uploader-icon"/>
  </el-upload>
</el-form-item>
```

```
<script lang="ts">
import {
  ElUploadInternalFileDetail,
  ElUploadInternalRawFile
} from 'element-ui/types/upload'

@Component({
  name: 'PlayerDetail'
})
export default class extends Vue {

  // 上传头像预览地址
  private imageUrl = '';

  // 头像上传成功
  private handleAvatarSuccess(res: any, file: ElUploadInternalFileDetail) {
    // 预览图片
    this.imageUrl = URL.createObjectURL(file.raw)
```

```
    // 将返回图片名称设置到playerForm上
    // 设置为响应中图片地址
    this.playerForm.avatar = file.name
  }

  private beforeAvatarUpload(file: ElUploadInternalRawFile) {
    const isLt2M = file.size / 1024 / 1024 < 1

    if (!isLt2M) {
      this.$message.error('上传头像图片大小不能超过1MB!')
    }
    return isLt2M
  }
}
</script>
```

样式设置

```
.avatar-uploader .el-upload {
  border: 1px dashed #d9d9d9;
  border-radius: 6px;
  cursor: pointer;
  position: relative;
  overflow: hidden;
}
.avatar-uploader .el-upload:hover {
  border-color: #409eff;
}
.avatar-uploader-icon {
  font-size: 28px;
  color: #8c939d;
  width: 178px;
  height: 178px;
  line-height: 178px;
  text-align: center;
}
.avatar {
  width: 178px;
  height: 178px;
  display: block;
}
```

校验

```
<script lang="ts">
import { Ref } from 'vue-property-decorator'
```

```
@Component({})
export default class extends Vue {

  // 设置引用
  @Ref('playerForm')
  private form!: Form;

  // 必填项校验函数，弹出一个message提示
  private validateRequire = (rule: any, value: string, callback: Function) =>
{
    if (value === '') {
      this.$message({
        message: rule.field + '必须填写',
        type: 'error'
      })

      callback(new Error(rule.field + '必须填写'))
    } else {
      callback()
    }
  };

  // 校验规则
  private rules = {
    accountname: [{ validator: this.validateRequire }],
    nickname: [{ validator: this.validateRequire }]
  };

  private submitForm() {
    this.form.validate(async valid => {
      if (valid) {
        // ...
      } else {
        console.error('校验失败，请修改后重试')
        return false
      }
    })
  }
}
</script>
```

玩家更新

路由注册, src/router/modules/players.ts

```
{
  path: 'edit/:id(\\d+)',
  component: () => import('@/views/player/edit.vue'),
    name: 'EditPlayer',
    meta: {
      title: 'editPlayer',
      noCache: true,
      activeMenu: '/players/list',
      hidden: true
    }
},
```

创建编辑页面，src/views/player/edit.vue

```
<template>
  <player-detail :is-edit="true" />
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator'
import PlayerDetail from './components/PlayerDetail.vue'

@Component({
  name: 'EditPlayer',
  components: {
    PlayerDetail
  }
})
export default class extends Vue {}
</script>
```

触发导航，

```
<el-table-column align="center" label="操作">
  <template v-slot="{row}">
    <router-link :to="'/players/edit/' + row.id">
      <el-button type="primary" icon="el-icon-edit" >
        更新
      </el-button>
    </router-link>
  </template>
</el-table-column>
```

删除玩家

弹窗确认，用户确认再删除。

```
<el-button
           type="danger"
           @click="handleDelete(scope)"
           >
  删除
</el-button>
```

```
private handleDelete(scope: any) {
  const { $index, row } = scope
  this.$confirm('确定删除玩家信息?', '提示', {
    confirmButtonText: '确定',
    cancelButtonText: '取消',
    type: 'warning'
  })
    .then(async() => {
    await deletePlayer(row.id)
    this.list.splice($index, 1)
    this.$message({
      type: 'success',
      message: '删除成功!'
    })
  })
    .catch(err => { console.error(err) })
}
```