link: null
title: 珠峰架构师成长计划
description: src/pages/User/_layout.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=177 sentences=601, words=4258

## 1. 生成项目

```
$ npm install dva-cli -g
$ dva -v
$ dva new zhufeng-dva-chat
```

## 2. 实现注册和登录

src/pages/User/_layout.js

```
import React,{Component} from 'react';
import styles from './User.less';
export default class UserLayout extends Component {
    render() {
        return (
            <div className={styles.container}>
                {this.props.children}
            div>
        )
    }
}
```

src/pages/User/User.less

```
.container{
    display: flex;
    height:100%;
    justify-content: center;
    align-items: center;
}
```

src/pages/User/Register.js

```
import React,{Component} from 'react';
import {connect} from 'dva';
import Link from 'umi/link';
import { Form, Input, Button, Select, Row, Col, Popover, Progress } from 'antd';
import styles from './Register.less';
const FormItem = Form.Item;
const { Option } = Select;
const InputGroup=Input.Group;

const passwordStatusMap = {
    ok: (
      <div className={styles.success}>
        强度：强
      div>
    ),
    pass: (
      <div className={styles.warning}>
        v强度：中
      div>
    ),
    poor: (
      <div className={styles.error}>
        强度：太短
      div>
    ),
};
const passwordProgressMap = {
    ok: 'success',
    pass: 'normal',
    poor: 'exception',
  };

@connect(({ register, loading }) => ({
    register,
    submitting: loading.effects['register/submit'],
  }))
@Form.create()
export default class extends Component {
    state = {
        count: 0,
        confirmDirty: false,
        visible: false,
        help: '',
        prefix: '86',
    };
    getPasswordStatus=() => {
        const { form } = this.props;
        const value = form.getFieldValue('password');
        if (value && value.length > 9) {
            return 'ok';
        }
        if (value && value.length > 5) {
            return 'pass';
        }
        return 'poor';
    };

    renderPasswordProgress = () => {
        const { form } = this.props;
    const value = form.getFieldValue('password');
    const passwordStatus = this.getPasswordStatus();
    return value && value.length ? (
      <div className={styles[`progress-${passwordStatus}`]}>
        <Progress
          status={passwordProgressMap[passwordStatus]}
          className={styles.progress}
          strokeWidth={6}
          percent={value.length * 10 > 100 ? 100 : value.length * 10}
          showInfo={false}
        />
      div>
    ) : null;
  };

  checkPassword = (rule, value, callback) => {
    const { visible, confirmDirty } = this.state;
    if (!value) {
      this.setState({
        help:"请输入密码!",
        visible: !!value,
      });
      callback('error');
    } else {
      this.setState({
        help: '',
      });
      if (!visible) {
        this.setState({
          visible: !!value,
        });
      }
      if (value.length < 6) {
        callback('error');
      } else {
        const { form } = this.props;
        if (value && confirmDirty) {
          form.validateFields(['confirm'], { force: true });
        }
        callback();
      }
    }
  };
checkConfirm = (rule, value, callback) => {
    const { form } = this.props;
    if (value && value !== form.getFieldValue('password')) {
      callback('两次输入的密码不匹配!');
    } else {
      callback();
    }
};
changePrefix = value => {
    this.setState({
      prefix: value,
    });
};
onGetCaptcha = () => {
    let count = 59;
    this.setState({ count });
    this.interval = setInterval(() => {
      count -= 1;
      this.setState({ count });
      if (count === 0) {
        clearInterval(this.interval);
      }
    }, 1000);
  };
render() {
    const { form, submitting } = this.props;
    const {getFieldDecorator}=form;
    const { count, prefix, help, visible } = this.state;
    return (
        <div className={styles.register}>
            <h3>用户注册h3>
            <Form onSubmit={this.handleSubmit}>
                <FormItem>
                    {getFieldDecorator('mail', {
                        rules: [
                            {
                            required: true,
                            message:  '邮箱必须输入',
                            },
                            {
                            type: 'email',
                            message: '邮箱格式不合法',
                            },
                        ],
                    })(
                    <Input size="large" placeholder="邮箱"/>
                    )}
                FormItem>
                <FormItem help={help}>
                    <Popover
                    content={
                        <div style={{ padding: '4px 0' }}>
                            {passwordStatusMap[this.getPasswordStatus()]}
                            {this.renderPasswordProgress()}
                            <div style={{ marginTop: 10 }}>
                              请至少输入 6 个字符。请不要使用容易被猜到的密码。
                            div>
                        div>
                    }
                    overlayStyle={{ width: 240 }}
                    placement="right"
                    visible={visible}
                    >
                    {getFieldDecorator('password', {
                        rules: [
                        {
                            validator: this.checkPassword,
                        },
                        ],
                    })(
                        <Input
                          size="large"
                          type="password"
                          placeholder='至少六位密码，区分大小写'
                        />
                    )}
                    Popover>
                FormItem>
            <FormItem>
                {getFieldDecorator('confirm', {
                rules: [
                    {
                     required: true,
                     message: '请输入确认密码！',
                    },
                    {
                     validator: this.checkConfirm,
                    },
                ],
                })(
                <Input
                    size="large"
                    type="password"
                    placeholder="确认密码"
                />
                )}
            FormItem>
            <FormItem>
                <InputGroup compact>
                <Select
                    size="large"
                    value={prefix}
                    onChange={this.changePrefix}
                    style={{ width: '20%' }}
                >
                    <Option value="86">+86Option>
                    <Option value="87">+87Option>
                Select>
                {getFieldDecorator('mobile', {
                    rules: [
                    {
                        required: true,
                        message: '请输入手机号!',
                    },
                    {
                        pattern: /^\d{11}$/,
                        message: '手机号格式错误！',
                    },
                    ],
                })(
                    <Input
                    size="large"
                    style={{ width: '80%' }}
                    placeholder='手机号'
                    />
                )}
                InputGroup>
            FormItem>
            <FormItem>
                <Row gutter={8}>
                    <Col span={16}>
                        {getFieldDecorator('captcha', {
                        rules: [
                            {
                            required: true,
                            message: '请输入验证码！',
                            },
                        ],
                        })(
                        <Input
                            size="large"
                            placeholder="验证码"
                        />
                        )}
                    Col>
                    <Col span={8}>
                        <Button
                        size="large"
                        disabled={count}
                        className={styles.getCaptcha}
                        onClick={this.onGetCaptcha}
                        >
                          {count? `${count} s`: '获取验证码'}
                        Button>
                    Col>
                Row>
            FormItem>
            <FormItem>
                <Button
                size="large"
                loading={submitting}
                className={styles.submit}
                type="primary"
                htmlType="submit"
                >
```

```
                    注册
                    Button>
                    <Link className={styles.login} to="/User/Login">
                      使用已有账户登录
                    Link>
                FormItem>
              Form>
            div>

        );
    }
}
```

src/pages/User/Register.less

```
.register{
    width:450px;
    border:1px solid #CCC;
    border-radius: 10px;
    padding:20px;
    h3{
        text-align: center;
    }
    .getCaptcha{
        width:100%;
        display:block;
    }
    .login{
        float:right;
    }
}
```

## 3. 实现手机验证码验证

```
import fetch from 'dva/fetch';

function parseJSON(response) {
  return response.json();
}

function checkStatus(response) {
  if (response.status >= 200 && response.status < 300) {
    return response;
  }

  const error = new Error(response.statusText);
  error.response = response;
  throw error;
}

function checkSuccess(data) {
  if (data.code == 0) {
    return data;
  }

  const error = new Error(data.error);
  throw error;
}

const HOST='http://localhost:3000';
export default function request(url, options) {
  return fetch(HOST+url, options)
    .then(checkStatus)
    .then(parseJSON)
    .then(checkSuccess)
    .catch(err => ({ err }));
}
```

src/pages/User/Register.js

```
-    onGetCaptcha = () => {
+    onGetCaptcha=() => {
+        const {dispatch,form}=this.props;
+        let mobile=form.getFieldValue('mobile');
+        dispatch({
+            type: 'register/getCaptcha',
+            payload:mobile
+        });
```

src/pages/User/models/register.js

```
import * as registerService from '../services/register';
export default {
  namespace: 'register',
  state: {

  },
  effects: {
      *getCaptcha({payload},{call,put}) {
        const {data} = yield call(registerService.getCaptcha,payload);
        console.log(data);
    }
  },
  reducers: {}
};
```

src/pages/User/services/register.js

```
import request from '../../../utils/request';
export function getCaptcha(mobile) {
  return request(`/user/getCaptcha?mobile=${mobile}`);
}
```

server/app.js

```
let express=require('express');
let session=require('express-session');
const cors=require('cors');
let app=express();
app.use(cors());
app.use(session({
    resave:true,
    secret:'zfpx',
    saveUninitialized:true
}));
let user=require('./routes/user');
app.use('/user',user);
app.listen(3000);
```

server/routes/user.js

```
let express=require('express');
let router=express.Router();
let axios=require('axios');
let {SMS}=require('../config');
router.get('/getCaptcha',async function (req,res) {
    let {mobile}=req.query;
    let captcha=new Date().getMilliseconds();
    req.session.captcha = captcha;
    let {data} = await axios({
        method: 'POST',
        url: 'https://open.ucpaas.com/ol/sms/sendsms',
        data: {
            ...SMS,
            mobile,
            param:captcha,
        },
        headers: {
            "Content-Type": "application/json;charset=utf-8",
            "Accept": "application/json"
        }
    });
    if (data.code == '000000') {
        res.json({code: 0,data:captcha});
    } else {
        res.json({code:1,error:'获取验证码失败!'});
    }
});
module.exports=router;
```

server/config.js

```
module.exports={
    SMS: {
        sid: '32548fb951ac0df279db0e6e9a515566@',
    token: 'aa0309c08920ca38201de69eb3c745b6@',
    appid: '16129d504b7c484c9e8f09b4ec929983@',
        templateid: '387675',
    }
}
```

## 4.手机注册和登录

server/app.js

```
+const bodyParser=require('body-parser');
 let app=express();
-app.use(cors());
+app.use(cors({
+    origin:'http://localhost:8000',
+    credentials:true
+}));
+app.use(bodyParser.json());
```

server/config.js

```
+    DB: {
+        url:'mongodb://localhost/chat'
+    }
```

server/models/index.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const ObjectId = Schema.Types.ObjectId;
const {
  DB
} = require('../config');
const conn = mongoose.createConnection(DB.url,{ useNewUrlParser: true });
const User = conn.model('User', {
    email: {type: String,required: true},
    password: {type: String,required: true},
    mobile:{type:String,required:true}
});
module.exports={
    User
}
```

server/routes/user.js

```
let express=require('express');
let router=express.Router();
let axios=require('axios');
let {SMS}=require('../config');
const {User}=require('../models');
router.get('/getCaptcha',async function (req,res) {
    let {mobile}=req.query;
    let captcha=new Date().getMilliseconds();
    req.session.mobile=mobile;
    req.session.captcha=captcha;
    let {data} = await axios({
        method: 'POST',
        url: 'https://open.ucpaas.com/ol/sms/sendsms',
        data: {
            ...SMS,
            mobile,
            param:captcha,
        },
        headers: {
            "Content-Type": "application/json;charset=utf-8",
            "Accept": "application/json"
        }
    });
    if (data.code == '000000') {
        res.json({code: 0,data:captcha});
    } else {

        res.json({code:1,error:'error'});
    }
});

router.post('/register',async function (req,res) {
    let {email,password,mobile,captcha}=req.body;
    if (req.session.captcha != captcha) {
        return res.json({code:0,error:'验证码错误!'});
    }
    let user=new User({email,password,mobile});
    try {
        await user.save();
        res.json({code:0,data:'注册成功'});
    } catch (error) {
        res.json({code:1,error});
    }
});
router.post('/login',async function (req,res) {
    let {email,password,captcha,type}=req.body;
    if (type=='mobile') {
        if (req.session.captcha != captcha) {
            return res.json({code:0,error:'登录失败!'});
        }
        let mobile=req.session.mobile;
        console.log(mobile,req.session.captcha , captcha);
        let oldUser=await User.findOne({mobile});
        if (oldUser) {
            req.session.user=oldUser;
            res.json({code:0,data:'登录成功'});
        } else {
            res.json({code:1,error:'用户名或密码错误'});
        }
    } else {
        let oldUser=await User.findOne({email,password});
        if (oldUser) {
            req.session.user=oldUser;
            res.json({code:0,data:'登录成功'});
        } else {
            res.json({code:1,error:'用户名或密码错误'});
        }
    }
});
module.exports=router;
```

src/pages/Rooms/index.jsx

```
import React,{Component} from 'react';
export default class Rooms extends Component{
    render(){
        return (
            <div>Roomsdiv>
        )
    }
}
```

src/pages/User/Login.jsx

```
import React,{Component} from 'react';
import {connect} from 'dva';
import Link from 'umi/link';
import {
    Form,
    Input,
    Button,
    Select,
    Row,
    Col,
    Popover,
    Progress,
    Tabs
} from 'antd';
import styles from './Login.less';
const FormItem=Form.Item;
const {Option}=Select;
const InputGroup=Input.Group;
const TabPane = Tabs.TabPane;

@connect(({register,loading}) => ({register,submitting: loading.effects['register/submit']}))
@Form.create()
export default class extends Component {
    state={
        count: 0,
        type:'account'
    };

    onGetCaptcha=() => {
        const {dispatch,form}=this.props;
        let mobile=form.getFieldValue('mobile');
        dispatch({type: 'register/getCaptcha',payload: mobile});
        let count=59;
        this.setState({count});
        this.interval=setInterval(() => {
            count-=1;
            this.setState({count});
            if (count===0) {
                clearInterval(this.interval);
            }
        },1000);
    };
    handleSubmit=e => {
        e.preventDefault();
        const {form,dispatch}=this.props;
        const {type}=this.state;
        form.validateFields({
            force: true
        },(err,values) => {
            if (!err) {
                dispatch({
                    type: 'login/submit',
                    payload: {
                        ...values,
                        type
                    }
                });
            }
        });
    };
    onSwitch = type => {
        this.setState({
          type,
        });
    };
    render() {
        const {form,submitting}=this.props;
        const {getFieldDecorator}=form;
        const {count}=this.state;
        return (

                用户注册

                            {getFieldDecorator('email',{
                                rules: [
                                    {
                                        required: true,
                                        message: '邮箱必须输入'
                                    },{
                                        type: 'email',
                                        message: '邮箱格式不合法'
                                    }
                                ]
                            })()}

                            {getFieldDecorator('password',{
                                rules: [
                                    {
                                        required: true,
                                        message: '请输入密码！'
                                    }
                                ]
                            })()}

...
```

src/pages/User/Login.less

```
.register{
    width:450px;
    border:1px solid #CCC;
    border-radius: 10px;
    padding:20px;
    h3{
        text-align: center;
    }
    .getCaptcha{
        width:100%;
        display:block;
    }
    .toRegister{
        float:right;
    }
}
```

src/pages/User/Register.jsx

```
import React,{Component} from 'react';
import {connect} from 'dva';
import Link from 'umi/link';
import {
    Form,
    Input,
    Button,
    Select,
    Row,
    Col,
    Popover,
    Progress
} from 'antd';
import styles from './Register.less';
const FormItem=Form.Item;
const {Option}=Select;
const InputGroup=Input.Group;

const passwordStatusMap={
    ok: (
        <div className={styles.success}>
            强度: 强
                div>
    ),
    pass: (
        <div className={styles.warning}>
            v强度: 中
                div>
    ),
    poor: (
        <div className={styles.error}>
            强度: 太短
                div>
    )
};
const passwordProgressMap={
    ok: 'success',
    pass: 'normal',
    poor: 'exception'
};

@connect(({register,loading}) => ({register,submitting: loading.effects['register/submit']}))
@Form.create()
export default class extends Component {
    state={
        count: 0,
        confirmDirty: false,
        visible: false,
        help: '',
        prefix: '86'
    };
    getPasswordStatus=() => {
        const {form}=this.props;
        const value=form.getFieldValue('password');
        if (value&&value.length>9) {
            return 'ok';
        }
        if (value&&value.length>5) {
            return 'pass';
        }
        return 'poor';
    };

    renderPasswordProgress=() => {
        const {form}=this.props;
        const value=form.getFieldValue('password');
        const passwordStatus=this.getPasswordStatus();
        return value&&value.length
            ? (
                <div className={styles[`progress-${passwordStatus}`]}>
                    <Progress
                        status={passwordProgressMap[passwordStatus]}
                        className={styles.progress}
                        strokeWidth={6}
                        percent={value.length*10>100
                            ? 100
                            :value.length*10}
                        showInfo={false} />
                div>
            )
            :null;
    };

    checkPassword=(rule,value,callback) => {
        const {visible,confirmDirty}=this.state;
        if (!value) {
            this.setState({
                help: "请输入密码!",
                visible: !!value
            });
            callback('error');
        } else {
            this.setState({help: ''});
            if (!visible) {
                this.setState({
                    visible: !!value
                });
            }
            if (value.length<6) {
                callback('error');
            } else {
                const {form}=this.props;
                if (value&&confirmDirty) {
                    form.validateFields(['confirm'],{force: true});
                }
                callback();
            }
        }
    };
    checkConfirm=(rule,value,callback) => {
        const {form}=this.props;
        if (value&&value!==form.getFieldValue('password')) {
            callback('两次输入的密码不匹配!');
        } else {
            callback();
        }
    };
    changePrefix=value => {
        this.setState({prefix: value});
    };
    onGetCaptcha=() => {
        const {dispatch,form}=this.props;
        let mobile=form.getFieldValue('mobile');
        dispatch({type: 'register/getCaptcha',payload: mobile});
        let count=59;
        this.setState({count});
        this.interval=setInterval(() => {
            count-=1;
            this.setState({count});
            if (count===0) {
                clearInterval(this.interval);
            }
        },1000);
    };
    handleSubmit=e => {
        e.preventDefault();
        const {form,dispatch}=this.props;
        form.validateFields({
            force: true
        },(err,values) => {
            console.log('values',values);
            if (!err) {
                const {prefix}=this.state;
                dispatch({
                    type: 'register/submit',
                    payload: {
                        ...values,
                        prefix
                    }
                });
            }
        });
    };
    render() {
        const {form,submitting}=this.props;
        const {getFieldDecorator}=form;
        const {count,prefix,help,visible}=this.state;
        return (
            <div className={styles.register}>
                <h3>用户注册h3>
                <Form onSubmit={this.handleSubmit}>
                    <FormItem>
                        {getFieldDecorator('email',{
                            rules: [
                                {
                                    required: true,
                                    message: '邮箱必须输入'
                                },{
                                    type: 'email',
                                    message: '邮箱格式不合法'
                                }
                            ]
                        })(<Input size="large" placeholder="邮箱" />)}
                    FormItem>
                    <FormItem help={help}>
                        <Popover
                            content={
                                <div style={{padding: '4px 0'}}>
                                    {passwordStatusMap[this.getPasswordStatus()]}
                                    {this.renderPasswordProgress()}
                                    <div style={{marginTop: 10}}>
                                      请至少输入 6 个字符。请不要使用容易被猜到的密码。
                                    div>
                                div>
                            }
                            overlayStyle={{width: 240}}
                            placement="right"
                            visible={visible}
                        >
```

        );

```
                            {getFieldDecorator('password',{
                                rules: [
                                    {
                                        validator: this.checkPassword
                                    }
                                ]
                            })(<Input size="large" type="password" placeholder='至少六位密码，区分大小写' />)}
                        Popover>
                    FormItem>
                    <FormItem>
                        {getFieldDecorator('confirm',{
                            rules: [
                                {
                                    required: true,
                                    message: '请输入确认密码！'
                                },{
                                    validator: this.checkConfirm
                                }
                            ]
                        })(<Input size="large" type="password" placeholder="确认密码" />)}
                    FormItem>
                    <FormItem>
                        <InputGroup compact>
                            <Select
                                size="large"
                                value={prefix}
                                onChange={this.changePrefix}
                                style={{
                                    width: '20%'
                                }}>
                                <Option value="86">+86Option>
                                <Option value="87">+87Option>
                            Select>
                            {getFieldDecorator('mobile',{
                                rules: [
                                    {
                                        required: true,
                                        message: '请输入手机号!'
                                    },{
                                        pattern: /^\d{11}$/,
                                        message: '手机号格式错误！'
                                    }
                                ]
                            })(<Input
                                size="large"
                                style={{
                                    width: '80%'
                                }}
                                placeholder='手机号' />)}
                        InputGroup>
                    FormItem>
                    <FormItem>
                        <Row gutter={8}>
                            <Col span={16}>
                                {getFieldDecorator('captcha',{
                                    rules: [
                                        {
                                            required: true,
                                            message: '请输入验证码！'
                                        }
                                    ]
                                })(<Input size="large" placeholder="验证码" />)}
                            Col>
                            <Col span={8}>
                                <Button
                                    size="large"
                                    disabled={count}
                                    className={styles.getCaptcha}
                                    onClick={this.onGetCaptcha}>
                                    {count
                                        ? `${count} s`
                                        :'获取验证码'}
                                Button>
                            Col>
                        Row>
                    FormItem>
                    <FormItem>
                        <Button
                            size="large"
                            loading={submitting}
                            className={styles.submit}
                            type="primary"
                            htmlType="submit">
                            注册
                                          Button>
                        <Link className={styles.login} to="/User/Login">
                            使用已有账户登录
                                          Link>
                    FormItem>
                Form>
            div>

        );
    }
}
```

src/pages/User/models/login.js

```
import * as loginService from '../services/login';
import router from 'umi/router';
export default {
  namespace: 'login',

  state: {

  },

  effects: {
      *getCaptcha({payload},{call,put}) {
         yield call(loginService.getCaptcha,payload);
    },
    *submit({payload},{call,put}) {
      let {code}=yield call(loginService.submit,payload);
      if(code == 0)
        router.push('/Rooms');
    }
  },

  reducers: {

  },
};
```

src/pages/User/models/register.js

```
import * as registerService from '../services/register';
import router from 'umi/router';
export default {
  namespace: 'register',

  state: {

  },

  effects: {
      *getCaptcha({payload},{call,put}) {
         yield call(registerService.getCaptcha,payload);
    },
    *submit({payload},{call,put}) {
      yield call(registerService.submit,payload);
      router.push('/User/Login');
    }
  },

  reducers: {

  },
};
```

src/pages/User/services/login.js

```
import request from '../../../utils/request';

export function getCaptcha(mobile) {
  return request(`/user/getCaptcha?mobile=${mobile}`);
}

export function submit(values) {
  return request(`/user/login`,{
    method: 'POST',
    body: JSON.stringify(values),
    headers: {
      'Content-Type': "application/json",
      "Accept":"application/json'"
    }
  });
}
```

src/pages/User/services/register.js

```
export function submit(values) {
  return request(`/user/register`,{
    method: 'POST',
    body: JSON.stringify(values),
    headers: {
      'Content-Type': "application/json",
      "Accept":"application/json'"
    }
  });
}
```

src/utils/request.js

```
-  return fetch(HOST+url, options)
+  return fetch(HOST+url,{
+    ...options,
+    credentials:'include'
+  })
```

## 5. 实现房间功能

server/app.js

```
+ let room=require('./routes/room');
app.use('/user',user);
+ app.use('/room',room);
```

server/models/index.js

```
+
+const Room = conn.model('Room', {
+    name: {type: String,required: true}
+});
+
 module.exports={
-    User
+    User,
+    Room
 }
```

server/routes/room.js

```
let express=require('express');
let router=express.Router();
const {Room}=require('../models');

router.get('/list',async function (req,res) {
    let list = await Room.find();
    res.json({
        code: 0,
        data:list
    });
});

router.post('/create',async function (req,res) {
    let {name}=req.body;
    await Room.create({name});
    res.json({
        code: 0,
        data:'房间创建成功'
    });
});

module.exports=router;
```

src/pages/Room/index.jsx

```
import React,{Component} from 'react';
import Link from 'umi/link';
import {connect} from 'dva';
import {Layout,Menu,Card,Input,Row,Col,Button,Form} from 'antd';
import styles from './index.less';
const {Header,Footer,Content}=Layout;

@connect(({room,loading}) => ({room,loading: loading.effects['room/list']}))
@Form.create()
export default class Rooms extends Component {
    createRoom=() => {
        let name=this.props.form.getFieldValue('keyword');
        if (name) {
            this.props.dispatch({
                type: 'room/create',
                payload:{name}
            });
        }
    }
    render() {
        let name=this.props.form.getFieldValue('keyword');
        let rooms=this.props.room.list;
        if (name) {
            rooms = rooms.filter(item => item.name.indexOf(name)!=-1);
        }

        return (
            <Layout>
                <Header>
                    <a className={styles.logo}>珠峰聊天室a>
                    <Menu
                        theme="dark"
                        mode="horizontal"
                        defaultSelectedKeys={['register']}
                        style={{ lineHeight: '64px' }}
                    >
                        <Menu.Item key="register">注册Menu.Item>
                        <Menu.Item key="login">登录Menu.Item>
                    Menu>
                Header>
                <Content>
                    <Card>
                        <Row gutter={16}>
                            <Col span={6} offset={8}>
                                {
                                    this.props.form.getFieldDecorator('keyword')(
                                        <Input type="text" placeholder="请输入房间名">Input>
                                    )
                                }
                            Col>
                        Row>
                    Card>
                    <Card>
                        <Row gutter={16}>
                            {
                                rooms.length>0?rooms.map(item => (
                                    <Col span={6} key={item._id}>
                                        <Card
                                            title={item.name}
                                            extra={<Link to={`/Rooms/${item._id}`}>进入Link>}
                                        >
                                            <p>用户1p>
                                            <p>用户2p>
                                            <p>用户3p>
                                        Card>
                                    Col>
                                )):<Button onClick={this.createRoom}>创建房间Button>
                            }
                        Row>
                    Card>
                Content>
                <Footer>
                    <div className={styles.copyright}> ©2018 珠峰培训div>
                Footer>
            Layout>
        )
    }
}
```

src/pages/Room/index.less

```
.logo{
    width:120px;
    height:64px;
    line-height: 64px;
    float:left;
}

.copyright{
    text-align:center;
}
```

src/pages/Room/models/room.js

```
import * as roomService from '../services/room';
export default {

  namespace: 'room',

  state: {
    list: []
  },

  subscriptions: {
    setup({ dispatch, history }) {
      history.listen(({pathname,query}) => {
        if(pathname == '/Room'){
          dispatch({
            type:'getRooms'
          });
        }
      });
    },
  },

  effects: {
    *getRooms({ payload }, { call, put }) {
      let {data: list}=yield call(roomService.getRooms);
      yield put({type: 'save',payload: {list}});
    },
    *create({ payload }, { call, put }) {
      yield call(roomService.create,payload);
      yield put({type: 'getRooms'});
    }
  },

  reducers: {
    save(state, action) {
      return { ...state, ...action.payload };
    },
  },
};
```

src/pages/Room/services/room.js

```
import request from '../../../utils/request';

export function getRooms() {
  return request(`/room/list`);
}
export function create(values) {
  return request(`/room/create`,{
    method: 'POST',
    body: JSON.stringify(values),
    headers: {
      'Content-Type': "application/json",
      "Accept":"application/json'"
    }
  });
}
```

## 6. 实现聊天功能

server/app.js

```
let express=require('express');
let session=require('express-session');
const cors=require('cors');
const bodyParser=require('body-parser');
const {User,Room,Message}=require('./models');
let app=express();
app.use(cors({
    origin:'http://localhost:8000',
    credentials:true
}));
app.use(bodyParser.json());
app.use(session({
    resave:true,
    secret:'zfpx',
    saveUninitialized:true
}));
let user=require('./routes/user');
let room=require('./routes/room');
app.use('/user',user);
app.use('/room',room);
let server=require('http').createServer(app);
server.listen(3000);
let io=require('socket.io')(server);
io.on('connection',async function (socket) {
    let userId=socket.handshake.query.user;
    let user=await User.findById(userId);
    let roomId=socket.handshake.query.room;

    let room=await Room.findById(roomId);
    socket.join(roomId);

    await User.findByIdAndUpdate(userId,{room: roomId});
    socket.on('getRoom',async function () {

        let users=await User.find({room: roomId});

        let messages = await Message.find({room: roomId});
        socket.emit('room',{room,users,messages});
    });
    socket.on('message',async function (content) {
        let {_id} = await Message.create({
            user:userId,
            content,
            room:roomId
        });
        let message=await Message.findById(_id).populate('user').populate('room').exec();
        io.in(roomId).emit('message',message);
    });

});
```

server/models/index.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const ObjectId = Schema.Types.ObjectId;
const {DB} = require('../config');
const conn = mongoose.createConnection(DB.url,{ useNewUrlParser: true });
const User = conn.model('User', {
    email: {type: String,required: true},
    password: {type: String,required: true},
    mobile: {type: String,required: true},
    avatar: {type: String,required: false},
    createAt: {type: Date,default: Date.now},
    room:{type:ObjectId,ref:'Room'}
});

const Room = conn.model('Room', {
    name: {type: String,required: true},
    avatar: {type: String},
    createAt:{type:Date,default:Date.now}
});

const Message = conn.model('Message', {
    content: {type: String},
    user: {type: ObjectId,ref: 'User'},
    createAt: {type: Date,default: Date.now},
    room:{type:ObjectId,ref:'Room'}
});

module.exports={
    User,
    Room,
    Message
}
```

server/routes/user.js

```
+ const gravatar=require('gravatar');
-    let user=new User({email,password,mobile});
+    let user=new User({email,password,mobile,avatar});
-            res.json({code:0,data:'登录成功'});
+            res.json({code:0,data:oldUser});
-            res.json({code:0,data:'登录成功'});
+            res.json({code:0,data:oldUser});
```

src/pages/Room/$id.jsx

```
import React,{Component} from 'react';
import {connect} from 'dva';
import {Layout,Menu,Card,Input,Row,Col,Button,Form,List,Avatar} from 'antd';
const {Header,Sider,Footer,Content}=Layout;
@connect(({chat}) => ({chat}))
@Form.create()
export default class Chat extends Component {
    talk=() => {
        let content=this.props.form.getFieldValue('content');
        this.props.dispatch({
            type: 'chat/talk',
            payload:content
        });
    }
    render() {
        let {users,messages}=this.props.chat;
        return (
            <Layout>
                <Header><a href="#">珠峰聊天室a>Header>
                <Layout>
                    <Sider>
                        <Menu
                            theme="dark"
                            mode="vertical"
                            defaultSelectedKeys={[]}
                        >
                            {
                                users.map(user => (<Menu.Item key={user._id}>{user.email}Menu.Item>))
                            }
                        Menu>
                    Sider>
                    <Content style={{height: '600px'}}>
                        <Card>
                            <List
                                itemLayout="horizontal"
                                dataSource={messages}
                                renderItem={item => (
                                <List.Item>
                                    <List.Item.Meta
                                        avatar={<Avatar src={item.user.avatar} />}
                                        title={item.user.email}
                                        description={item.content}
                                    />
                                List.Item>
                            )}
                         />
                        Card>
                        <Card>
                            <Row gutter={8}>
                                <Col span={23}>
                                    {
                                        this.props.form.getFieldDecorator('content')(<Input/>)
                                    }
                                Col>
                                <Col span={1}><Button onClick={this.talk}>发言Button>Col>
                            Row>
                        Card>
                    Content>
                Layout>
                <Footer style={{textAlign:'center'}}>
                        @2018 珠峰培训
                Footer>
            Layout>
        )
    }
}
```

src/pages/Room/index.jsx

```
-     extra={进入}
+     extra={进入}
```

src/pages/Room/models/chat.js

```
import IO from 'socket.io-client';
import router from 'umi/router';
export default {
  namespace: 'chat',

  state: {
    room: {},
    users: [],
    messages:[]
  },

  subscriptions: {
    setup({ dispatch, history }) {
      history.listen(({pathname,query}) => {
        let regexp=/\/Room\/([^/]+)$/;
        let result=pathname.match(regexp);
        if (result) {
          let roomId=result[1];
          let userStr = window.sessionStorage.getItem('user');
          if (userStr) {
            let user=JSON.parse(userStr);
            const socket= window.socket = new IO('http://localhost:3000',{query:{user:user._id,room:roomId}});
            socket.on('room',function ({room,users,messages}) {
                dispatch({type: 'save',payload: {room,users,messages}});
            });
            socket.emit('getRoom');
            socket.on('message',function (message) {
              console.log(message);
              dispatch({type: 'messageAdded',payload:message});
            });
          } else {
            router.push('/User/Login');
          }
        }
      });
    },
  },
  effects: {
    *talk({payload},{call,put,select}) {
      window.socket.emit('message',payload);
   }
  },
  reducers: {
    save(state, action) {
      return { ...state, ...action.payload };
    },
    messageAdded(state,{payload}) {
      return {...state, messages:[...state.messages,payload] };
    }
  }
};
```

src/pages/Room/models/room.js

```
-    dispatch({
-       type:'getRooms'
-    });
+    dispatch({type:'getRooms'});
```

src/pages/User/models/login.js

```
-      let {code}=yield call(loginService.submit,payload);
-      if(code == 0)
-        router.push('/Rooms');
+      let {code,data}=yield call(loginService.submit,payload);
+      if (code==0) {
+        yield put({type: 'save',payload: {user: data}});
+        window.sessionStorage.setItem('user',JSON.stringify(data));
+        router.push('/Room');
+      }
```

## 7. 上传图片

server/app.js

```
-app.use(bodyParser.json());
+app.use(bodyParser.json({limit:'1000kb'}));

-        let messages = await Message.find({room: roomId});
-        console.log(room,users,messages);
+        let messages = await Message.find({room: roomId}).populate('user');
```

server/models/index.js

```
-    createAt:{type:Date,default:Date.now}        //创建时间
+    createAt: {type: Date,default: Date.now},        //创建时间
+    users:[{type:ObjectId,ref:'User'}]
```

server/routes/room.js

```
-const {Room}=require('../models');
+const {Room,User}=require('../models');

-    let list = await Room.find();
+    let list=await Room.find();
+    for (let i=0;i
+        let users=await User.find({room: list[i]._id});
+        console.log(users);
+        list[i].users=users;
+    }
+    console.log(list);
```

server/routes/user.js

```
+ router.post('/changeAvatar',async function (req,res) {
+     let {userId,avatar}=req.body;
+     try {
+         let user = await User.findById(userId);
+         user.avatar=avatar;
+         await user.save();
+         res.json({code:0,data:user});
+     } catch (error) {
+         res.json({code:1,error});
+     }
+ });
```

src/pages/Room/index.jsx

```
import React,{Component} from 'react';
import Link from 'umi/link';
import {connect} from 'dva';
import Cropper from 'react-cropper';
import 'cropperjs/dist/cropper.css';
import {Layout,Menu,Card,Input,Row,Col,Button,Form,List,Avatar,Modal} from 'antd';
import styles from './index.less';
const {Header,Footer,Content}=Layout;

@connect(({room,login,loading}) => ({room,login}))
@Form.create()
export default class Rooms extends Component {
    state={
        src: '',
        cropperVisible: false
    }
    createRoom=() => {
        let name=this.props.form.getFieldValue('keyword');
        if (name) {
            this.props.dispatch({
                type: 'room/create',
                payload:{name}
            });
        }
    }

    changeAvatar=(userId) => {
        let that=this;
        const $input=document.createElement('input');
        $input.style.display = 'none';
        $input.setAttribute('type', 'file');
        $input.setAttribute('accept','*/*');
        $input.onchange=(e) => {
            const file=e.target.files[0];
            if (!file) return;
            const reader=new FileReader();
            reader.onloadend=function () {
                let avatar=this.result;
                that.setState({
                    src: avatar,
                    cropperVisible:true
                });

            }
            reader.readAsDataURL(file);
        }
        $input.click();
    }
    confirmChange=(userId) => {
        let avatar = this.cropper.getCroppedCanvas().toDataURL();
        this.setState({cropperVisible:false});
        this.props.dispatch({
              type: 'login/changeAvatar',
              payload: {userId,avatar}
        });
    }
    render() {
        let name=this.props.form.getFieldValue('keyword');
        let rooms=this.props.room.list;
        if (name) {
            rooms = rooms.filter(item => item.name.indexOf(name)!=-1);
        }

        return (
            <Layout>
                <Header>
                    <Row>
                        <Col span={2}><a className={styles.logo}>珠峰聊天室a>Col>
                        <Col span={21}><Menu
                        theme="dark"
                        mode="horizontal"
                        defaultSelectedKeys={['register']}
                        style={{ lineHeight: '64px' }}
                        >
                        <Menu.Item key="register">注册Menu.Item>
                        <Menu.Item key="login">登录Menu.Item>
                        Menu>Col>
                        <Col span={1}>{this.props.login.user&&<Avatar onClick={()=>this.changeAvatar(this.props.login.user._id)} src=
{this.props.login.user.avatar}/>}Col>
                    Row>
                Header>
                <Content>
                    <Card>
                        <Row gutter={16}>
                            <Col span={6} offset={8}>
                                {
                                    this.props.form.getFieldDecorator('keyword')(
                                        <Input type="text" placeholder="请输入房间名">Input>
                                    )
                                }
                            Col>
                        Row>
                    Card>
                    <Card>
                        <Row gutter={16}>
                            {
                                rooms.length>0?rooms.map(item => (
                                    <Col span={6} key={item._id}>
                                        <Card
                                            title={item.name}
                                            extra={<Link to={`/Room/${item._id}`}>进入Link>}
                                        >
                                            {
                                                item.users.map(user => <Avatar style={{marginLeft:5}} src={user.avatar} />)
                                            }
                                        Card>
                                    Col>
                                )):<Button onClick={this.createRoom}>创建房间Button>
                            }
                        Row>
                    Card>
                Content>
                <Footer>
                    <div className={styles.copyright}> ©2018 珠峰培训div>
                Footer>
                <Modal
                    onOk={()=>this.confirmChange(this.props.login.user._id)}
                    onCancel={()=>this.setState({cropperVisible:false})}
                    visible={this.state.cropperVisible} >
                    <Cropper
                            ref={i=>this.cropper =i}
                            src={this.state.src}
                            style={{height: 400, width: 400}}
                            aspectRatio={16 / 9}
                        guides={false} />
                Modal>
            Layout>
        )
    }
}
```

src/pages/Room/models/chat.js

```
+ dispatch({type: 'login/save',payload: {user}});
```

src/pages/Room/models/room.js

```
-        if(pathname == '/Room'){
-          dispatch({type:'getRooms'});
+        if (pathname=='/Room') {
+          let userStr = window.sessionStorage.getItem('user');
+          if (userStr) {
+            let user=JSON.parse(userStr);
+            dispatch({type: 'login/save',payload: {user}});
+            dispatch({type:'getRooms'});
+          } else {
+            router.push('/User/Login');
+          }
```

src/pages/User/models/login.js

```
+    *changeAvatar({payload},{call,put}) {
+      let {code,data}=yield call(loginService.changeAvatar,payload);
+      if (code==0) {
+        window.sessionStorage.setItem('user',JSON.stringify(data));
+        yield put({type: 'save',payload: {user: data}});
+      }
```

src/pages/User/services/login.js

```
export function changeAvatar(values) {
  return request(`/user/changeAvatar`,{
    method: 'POST',
    body: JSON.stringify(values),
    headers: {
      'Content-Type': "application/json",
      "Accept":"application/json'"
    }
  });
}
```

## 参考链接