

项企平台代码走查

REVIEWER: 彭玲 TIME: 2022/5/5

项企平台代码走查

总览

FS-EMIS 代码走查

1. api 侧

npm 包

启动参数

代码不干净

中间件

数据库访问

代码内敛性

代码逻辑性

日志等级

日志记录 (`console.log`)

2. web 侧

启动参数

未使用代码

避免 `console.log`

命名规范性

workflow 管理

FS-PEP 代码走查 (Web 侧)

package 包过时

无用代码

无用的方法

无用的注释

命名规范性

函数名

参数名

代码逻辑

FS-ProjectManagement 代码走查

1. api 侧

npm 包

其他

2. web 侧

启动参数

未使用代码

`console.log`

FS-OA 代码走查 (Web 侧)

启动参数

未使用代码

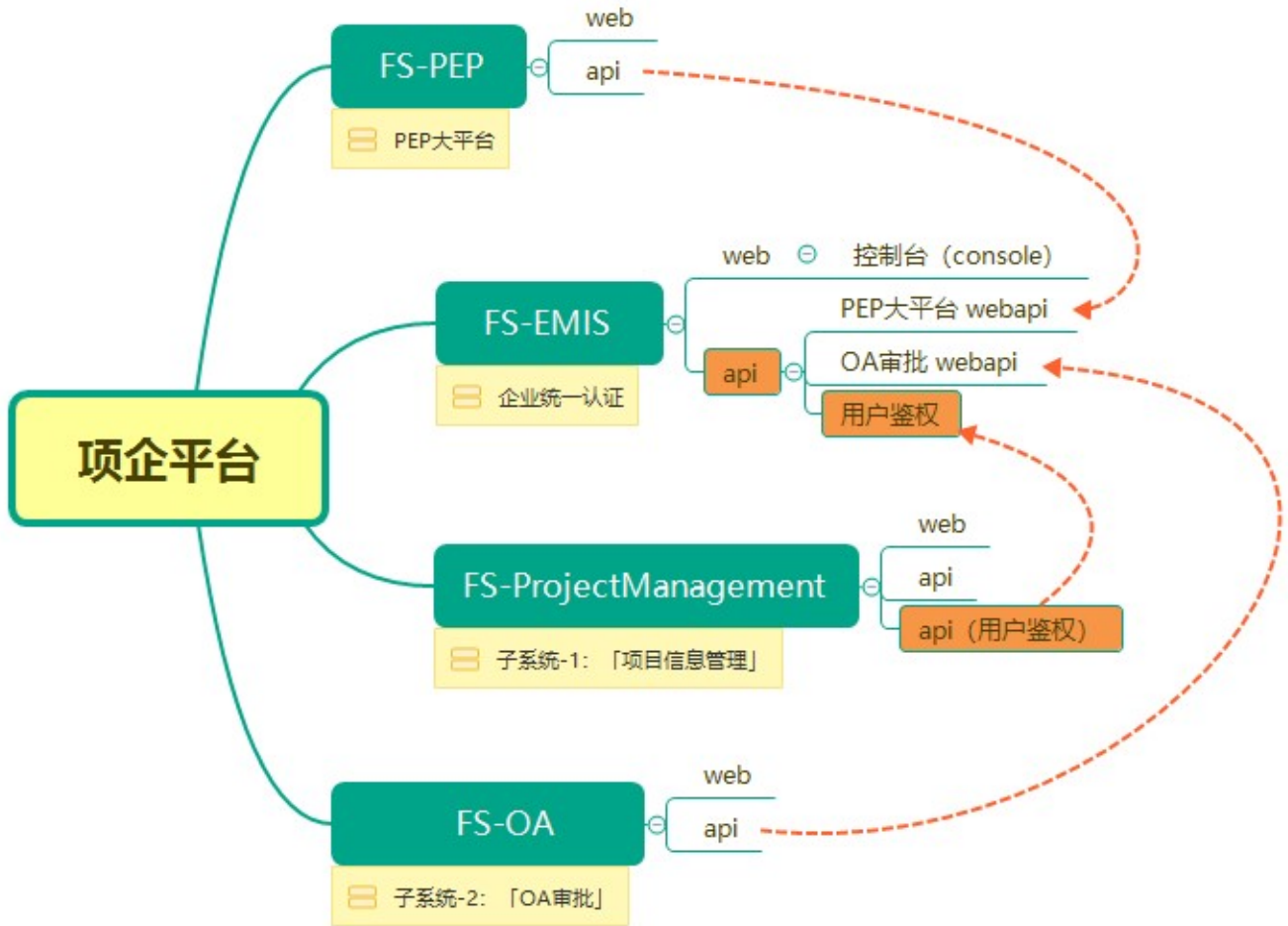
避免 `console.log`

总览

项企平台「项目」及「网站」对应关系如下表所示。

项目 (代码)	网站	备注
FS-PEP	PEP 大平台	
FS-EMIS	CA 企业认证	- 企业统一认证 - PEP 「审批中心」
FS-OA	「OA审批」子系统	
FS-ProjectManagement	「项目信息管理」子系统	

项企平台涉及项目及关系如下图所示。



FS-EMIS 代码走查

FS-EMIS 包含 Web 侧和 api 侧，SVN: <http://10.8.30.22/FS-EMIS/trunk/codes>，其中，

- Web 侧: CA 企业统一认证控制台 (console)
- api 侧: FS-PEP 平台的 api、FS-OA 子系统的 api、FS-ProjectManagement 用户鉴权 api

```
1 | FS-EMIS\trunk\codes
2 | |— api
3 | |— web
```

1. api 侧

npm 包

- `sequelize-automate` 包可以归于 `devDependencies` 中。
- `koa-convert` 不再需要。

启动参数

代码中只对以下参数做了检查，其他如 `FS_EMIS_WEB` 等使用时未检查是否合理（不为空），对此，可以对参数做初始化。

```
1 | if (!FS_UNIAPP_DB || !FS_CAMUNDA_HOST || !FS_CAMUNDA_ROOT || !PEP_KAFKA_BROKERS) {
2 |     // || !IOTA_REDIS_SERVER_HOST || !IOTA_REDIS_SERVER_PORT ||
   |     !IOTA_REDIS_SERVER_PWD) {
3 |     console.log('缺少启动参数, 异常退出');
4 |     args.showHelp();
5 |     process.exit(-1);
6 | }
```

代码不干净

- `sequelize-automate.config.js.js` 未被使用。
- `package-lock.json` 必需？不必需不提交 SVN。
- `utils/forward-api.js` 未被使用。
- `static/` 下的 `files` 和 `upload` 需要提交 SVN？
- `models/` 下自动生成的模型未被使用。
- 去掉不需要的包引用代码，如下面的 `app/lib/index.js`。

```
1 | // app/lib/index.js
2 | const apiLog = require('./middlewares/api-log');
3 | const kafka = require('./middlewares/kafka');
4 | module.exports.entry = function (app, router, opts) {
5 |     // app.use(kafka(app, opts));
6 |     // router.use(apiLog(app, opts));
7 | }
```

中间件

是否真的需要作为中间件，如 `business-rest.js` 等。

```
1 | // app/lib/middlewares/business-rest.js
2 |
3 | 'use strict';
4 |
5 | const request = require('superagent');
```

```

6  const buildurl = (url,token) => {
7      let connector = url.indexOf('?') === -1 ? '?' : '&';
8      return `${url}${connector}token=${token}`;
9  };
10
11  function factory(app, router, opts) {
12      return async function (ctx, next) {
13
14          const token = ctx.fs.api.token;
15
16          //console.log(username,password)
17          const req = {
18              get: (url, query) => {
19                  return request
20                      .get(buildurl(url,token))
21                      .query(query)
22              },
23              post: (url, data, query) => {
24                  return request
25                      .post(buildurl(url,token))
26                      .query(query)
27                      //.set('Content-Type', 'application/json')
28                      .send(data);
29              },
30
31              put: (url, data) => {
32                  return request
33                      .put(buildurl(url,token))
34                      //.set('Content-Type', 'application/json')
35                      .send(data);
36              },
37
38              delete: (url) => {
39                  return request
40                      .del(buildurl(url,token))
41              },
42          };
43
44          app.business = app.business || {};
45          app.business.request = req;
46
47          await next();
48      };
49  }
50
51  module.exports = factory;

```

改为:

```

1  'use strict';
2
3  const request = require('superagent');
4

```

```

5  const buildUrl = (url, token) => {
6      let connector = url.indexOf('?') === -1 ? '?' : '&';
7      return `${url}${connector}token=${token}`;
8  };
9
10 function factory(app, router, opts) {
11     let token = null;
12
13     const req = {
14         get: (url, query) => {
15             return request
16                 .get(buildUrl(url, token))
17                 .query(query)
18         },
19         post: (url, data, query) => {
20             return request
21                 .post(buildUrl(url, token))
22                 .query(query)
23                 .send(data);
24         },
25
26         put: (url, data) => {
27             return request
28                 .put(buildUrl(url, token))
29                 .send(data);
30         },
31
32         delete: (url) => {
33             return request
34                 .del(buildUrl(url, token))
35         },
36     };
37
38     app.business = app.business || {};
39     app.business.request = req;
40
41     return async function (ctx, next) {
42         token = ctx.fs.api.token;
43         await next();
44     };
45 }
46
47 module.exports = factory;

```

数据库访问

如登录接口 /login 实现中，在 `userRes_.length > opts.maxLogNum` 之前的条件不满足时，再查询 `UserToken` 数据库表。

```

1  const userRes = await models.User.findOne({
2      where: {
3          account: params.username,
4          password: password,

```

```

5     delete: false,
6   },
7   attributes: { exclude: ['password'] },
8 });
9
10 let where = {
11   expired: {
12     $between: [
13       moment().format('YYYY-MM-DD HH:mm:ss'),
14       moment().add(opts.expiration, 'minutes').format('YYYY-MM-DD HH:mm:ss')
15     ]
16   }
17 }
18 let findObj = {
19   where,
20   attributes: ['token', 'expired'],
21 }
22 const userRes_ = await models.UserToken.findAll(
23   findObj
24 );
25
26 if (!userRes) {
27   ctx.status = 400;
28   ctx.body = {
29     "message": "账号或密码错误"
30   }
31 } else if (!userRes.state) {
32   ctx.status = 400;
33   ctx.body = {
34     "message": "该用户已被禁用"
35   }
36 } else if (userRes.lock) {
37   ctx.status = 400;
38   ctx.body = {
39     "message": "该用户已被锁定"
40   }
41 } else if (userRes_.length > opts.maxLogNum) { // 这里查询数据库表 UserToken
42   ctx.status = 400;
43   ctx.body = {
44     "message": `登录人数限制在${opts.maxLogNum}`
45   }
46 } else {
47   ...
48 }

```

代码内敛性

如登录接口 `/login` 实现中包含：功能权限查询、数据范围查询、所在部门查询、删除 token 等功能。

代码逻辑性

以下代码中，`if()` 判断条件恒成立，直接更新 `expired` 即可。

```

1 // app/lib/middlewares/authenticator.js
2
3 let authorizeToken = async function (ctx, token, opts) {
4     const axyRes = await ctx.fs.dc.models.UserToken.findOne({
5         where: {
6             token: token,
7             expired: { $gte: moment().subtract(opts.expiration,
8 'minutes').format('YYYY-MM-DD HH:mm:ss') }
9         }
10    });
11    const { userInfo, expired } = axyRes;
12    if (!expired || moment().subtract(opts.expiration, 'minutes').valueOf() <=
13 moment(expired).valueOf()) { // 判断条件恒成立
14        await ctx.fs.dc.models.UserToken.update({
15            expired: moment(expired).add(opts.expiration, 'minutes').format()
16        }, {
17            where: {
18                token: token,
19            }
20        });
21        rs1t = {
22            'authorized': userInfo.authorized,
23            'resources': (userInfo || {}).resources || [],
24        };
25    }
26 }

```

日志等级

`config.js` 中定义了日志等级：开发环境下，日志等级为 `debug`；生产环境下，日志等级为 `info`。

建议接口中日志改用 `debug` 等级，仅开发环境做日志记录即可。

```

1 ctx.fs.logger.info(`[/login] START Time---: ${t0.format('YYYY-MM-DD HH:mm:ss.SSS')}`);
2
3 ...
4
5 ctx.fs.logger.info(`[/login] END Time---: ${t11.format('YYYY-MM-DD HH:mm:ss.SSS')}--
6 Total Time: [`, t11.valueOf() - t0.valueOf(), 'ms] ');

```

日志记录 (console.log)

以下日志记录使用 `console.log`，应该改用 `ctx.fs.logger.error()`。

```

1 // app/lib/controllers/auth/index.js
2 function getRedisInfo(key, ctx) {
3     return new Promise((resolve, reject) => {
4         if (ctx.redis) {
5             ctx.redis.get(key, function (err, result) {
6                 if (err) {
7                     console.log("请求redis失败: ", err); // 控制台日志
8                     return null;
9                 }
10            });
11        }
12    });
13 }

```

```

9         }
10        resolve(result);
11    })
12    } else {
13        resolve();
14    }
15 })
16 }

```

2. web 侧

启动参数

对启动参数赋初值，而非空值判断。

```

1  // // 启动参数
2  args.option(['p', 'port'], '启动端口');
3  args.option(['u', 'api-url'], 'webapi的URL');
4  args.option(['c', 'camundaHost'], 'camunda rest host');
5  args.option(['r', 'camundaRoot'], 'camunda rest root');
6  args.option('qnak', 'qiniuAccesskey');
7  args.option('qnsk', 'qiniuSecretKey');
8  args.option('qnbkt', 'qiniuBucket');
9  args.option('qndmn', 'qiniuDomain');
10 args.option(['d', 'domain'], 'web domain');
11 args.option('pepwebUrl', '大平台web的URL');
12 args.option('emiswebUrl', '统一认证web的URL');
13
14 const flags = args.parse(process.argv);
15
16 const FS_UNIAPP_API = process.env.FS_UNIAPP_API || flags.apiUrl;
17
18 const FS_CAMUNDA_HOST = process.env.CAMUNDA_HOST || flags.camundaHost;
19 const FS_CAMUNDA_ROOT = process.env.CAMUNDA_ROOT || flags.camundaRoot;
20 const ANXINCLOUD_QINIUI_ACCESSKEY = process.env.ANXINCLOUD_QINIUI_ACCESSKEY ||
  flags.qnak;
21 const ANXINCLOUD_QINIUI_SECRETKEY = process.env.ANXINCLOUD_QINIUI_SECRETKEY ||
  flags.qnsk;
22 const ANXINCLOUD_QINIUI_BUCKET_RESOURCE = process.env.ANXINCLOUD_QINIUI_BUCKET_RESOURCE
  || flags.qnbkt;
23 const ANXINCLOUD_QINIUI_DOMAIN_QNDMN_RESOURCE =
  process.env.ANXINCLOUD_QINIUI_DOMAIN_QNDMN_RESOURCE || flags.qndmn;
24 const FS_PEP_DOMAIN = process.env.FS_PEP_DOMAIN || flags.domain;
25 const FS_PEP_WEB_URL = process.env.FS_PEP_WEB_URL || flags.pepwebUrl;
26 const FS_EMIS_WEB_URL = process.env.FS_EMIS_WEB_URL || 'http://localhost:5000';
27
28 if (!FS_UNIAPP_API || !FS_CAMUNDA_HOST || !FS_CAMUNDA_ROOT ||
  !ANXINCLOUD_QINIUI_ACCESSKEY || !ANXINCLOUD_QINIUI_SECRETKEY ||
  !ANXINCLOUD_QINIUI_BUCKET_RESOURCE || !ANXINCLOUD_QINIUI_DOMAIN_QNDMN_RESOURCE
29     || !FS_PEP_DOMAIN
30 ) {
31     console.log('缺少启动参数, 异常退出');
32     args.showHelp();

```



```
33 process.exit(-1);
34 }
```

未使用代码

- `client/src/sections/struct` 未被使用。
- `routes/attachment/index.js -> getDomain()` 未被使用。

避免 `console.log`

生产环境下, 不用 `console.log`, 如 `client/src/sections/workflow/containers/editor-form/scalable-form-antd/xformCorrelationFormField/index.js` 代码。

```
1 console.log(RouteTable.getDomain);
```

命名规范性

变量、函数 等命名应该做到 见名知意。

- `config.js -> FS_UNIAPP_API` 中 `UNIAPP` 代表什么? 统一认证平台。是否改用 `EMIS` 更合适?

workflows管理

应用管理界面:



总数统计实现代码如下:

```
1 // client/src/sections/workflow/containers/application.js
2
3 initworkflowGroups = (appId) => {
4   this.props.dispatch(getworkflowGroups(appId)).then(res => {
5     if (res.success) {
6       //统计分类下流程的数量
7       let workflowGroupNum = 0;
8       let workflowCount = [];
9       res.payload.data.map(workflowGroup => {
10        let workflowNum = 0;
11        workflowGroup.child.map(workflow => {
12          workflowNum = workflowNum + workflow.workflowProcesses.length
```

```

13         workflowGroupNum = workflowGroupNum +
workflow.workflowProcesses.length
14         })
15         workflowCount.push({ id: workflowGroup.id, count: workflowNum });
16     })
17     workflowCount.push({ id: 0, count: workflowGroupNum });
18     this.setState({ workflowCount: workflowCount })
19     }
20     });
21 }
22
23 <TabPane tab={`全部(${workflowCount.filter(c => c.id == 0)[0] ? workflowCount.filter(c
=> c.id == 0)[0].count : 0)}`} key="all">
24 ...
25 </TabPane>

```

优化方案：后端做计数统计，且限制“全部”展示的个数，比如一次展示30个，分批展示剩余内容。

FS-PEP 代码走查 (Web 侧)

FS-PEP (大平台) 代码仅 Web 侧，SVN: <http://10.8.30.22/FS-PEP/trunk/codes/web>

package 包过时

koa-proxy 升级后 (1.0.0-alpha.3) , 不再需要 koa-convert 包。

无用代码

无用的方法

如，auth 模块的 useEffect() 方法。

```

1 // client/src/sections/auth/containers/login.js
2
3     useEffect(() => {
4
5     }, [])
6
7     useEffect(() => {
8         if (error) {
9             message.error(error);
10            setPassword('')
11        }
12    }, [error])

```

无用的注释

如 homepage 模块 containers 中如下大片注释，可以去掉。

```

1 // PEP首页的“待办事项”、“最新单位公告”定时

```

```

2 // homepageInterval = setInterval(() => {
3 //   if (userId) {
4 //     dispatch(getCompanyNotice(userId)).then(res => {
5 //       setauditUnit(res.payload.data.length)
6 //       setShowCompanyNotice(res.payload.data);
7 //     })
8 //   }
9 //   dispatch(getMyAuditList(params)).then(res => {
10 //     setauditList(res.payload.data.slice(0, 5))
11 //     setauditCunt(res.payload.data.length)
12 //     setHomeData(false)
13 //   })
14 // }, 60000)

```

命名规范性

函数名

例如，auth 模块 `getIsLogout()` 命名。

```

1 const getIsLogout = () => {
2   let search = window.location.search
3   let hasIsLogout = search.includes('isLogout')
4   return hasIsLogout ? search.match(/isLogout=(.*)/)[1] : null
5 }

```

又如，homepage 模块 `renderFunc()` 本身就是 `function`，没必要再加 `Func`，应该体现要 `render` 什么。

```

1 const renderFunc = (res, url, code) => { ... }

```

参数名

homepage 模块，参数名 `v` 不明晰，另外，`url` 可以直接作为参数，无需对象化。

```

1 const jumpSystem = (v, code) => {
2   ...
3   window.open(v.url);
4   ...
5 }
6 // 调用
7 jumpSystem({ url }, code)

```

代码逻辑

auth 模块 reducers 中 `user` 初始化为空对象的意义是什么？

```
1 const initState = {
2   requestUpdate: false, //是否请求用户、部门等数据依据
3   user: {},
4   isRequesting: false,
5   error: null
6 };
```

FS-ProjectManagement 代码走查

FS-ProjectManagement (子系统) 包含 Web 侧和 api 侧, SVN: <http://10.8.30.22/FS-ProjectManagement/trunk>

```
1 FS-ProjectManagement\trunk
2 |— web
3 |— web-api
```

1. api 侧

npm 包

- `sequelize-automate` 包可以归于 `devDependencies` 中。
- 升级 `koa-proxy` 至最新, 升级 `koa-proxy` 后 `koa-convert` 不再需要。

其他

其他见: `FS-EMIS 代码走查` -> 1. `api` 侧。

2. web 侧

启动参数

对启动参数赋初值, 而非空值判断。

```
1 // 启动参数
2 args.option(['p', 'port'], '启动端口');
3 args.option(['u', 'api-url'], 'webapi的URL');
4 args.option(['t', 'ty-api-url'], '统一管理平台webapi的URL');
5 args.option(['f', 'ty-web-url'], '统一管理平台web的URL');
6 args.option('pepwebUrl', '项企大平台web的URL');
7 args.option('pmApiUrl', '项目管理api的URL'); //商用环境用域名时配置该参数
8 args.option('qnak', 'qiniuAccesskey');
9 args.option('qnsk', 'qiniuSecretkey');
10 args.option('qnbkt', 'qiniuBucket');
11 args.option('qndmn', 'qiniuDomain');
12 args.option(['d', 'domain'], 'web domain');
13
14 const flags = args.parse(process.argv);
15 const FS_PROJECTMGT_API = process.env.FS_PROJECTMGT_API || flags.apiUrl;
16 const FS_TYMG_API = process.env.FS_TYMG_API || flags.tyApiUrl;
17 const FS_TYMG_WEB = process.env.FS_TYMG_WEB || flags.tywebUrl;
```

```

18 const FS_PEP_WEB = process.env.FS_PEP_WEB || flags.pepwebUrl;
19 const FS_PM_API = process.env.FS_PM_API || flags.pmApiUrl;
20 const ANXINCLOUD_QINIU_ACCESSKEY = process.env.ANXINCLOUD_QINIU_ACCESSKEY ||
  flags.qnak;
21 const ANXINCLOUD_QINIU_SECRETKEY = process.env.ANXINCLOUD_QINIU_SECRETKEY ||
  flags.qnsk;
22 const ANXINCLOUD_QINIU_BUCKET_RESOURCE = process.env.ANXINCLOUD_QINIU_BUCKET_RESOURCE
  || flags.qnbkt;
23 const ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE =
  process.env.ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE || flags.qndmn;
24 const FS_PEP_DOMAIN = process.env.FS_PEP_DOMAIN || flags.domain;
25
26 if (!FS_PROJECTMGT_API || !FS_TYMG_API || !FS_TYMG_WEB || !FS_PEP_WEB ||
  !ANXINCLOUD_QINIU_ACCESSKEY || !ANXINCLOUD_QINIU_SECRETKEY ||
27   !ANXINCLOUD_QINIU_BUCKET_RESOURCE || !ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE ||
  !FS_PEP_DOMAIN) {
28   console.log('缺少启动参数, 异常退出');
29   args.showHelp();
30   process.exit(-1);
31 }

```

未使用代码

sections/auth 模块、sections/approvalCenter 未被使用，可去掉。

```

1 // client/src/app.js
2
3 let sections = [
4   Auth,
5   personalDetail,
6   ProjectManage,
7   ConstructionLog,
8   ProjectApprove,
9   DocumentCenter,
10  OrganizationManage,
11  clientManage,
12  clientVisit,
13  reserveProjectManage,
14  ProjectAlternatives,
15  // approvalCenter,
16  //暂时隐藏
17  quotationManage,
18  quotationManageSales
19 ];

```

console.log

代码中很多 console.log 及其注释。

```

1 // client/src/layout/containers/layout/index.js
2 renderItem = (route, params, routes, paths) => {
3   // console.log(route, params, routes, paths)
4   ...
5   console.log(paths_)
6 }

```

FS-OA 代码走查 (Web 侧)

FS-OA (子系统) 代码仅 Web 侧, SVN: <http://10.8.30.22/FS-OA/trunk/web>

启动参数

对启动参数赋初值, 而非空值判断。

```

1 // // 启动参数
2 args.option(['p', 'port'], '启动端口');
3 args.option(['u', 'api-url'], 'webapi的URL');
4 args.option(['c', 'camundaHost'], 'camunda rest host');
5 args.option(['r', 'camundaRoot'], 'camunda rest root');
6 args.option('pepwebUrl', '项企大平台web的URL');
7 args.option('emiswebUrl', '企业统一认证管理平台web的URL');
8 args.option(['d', 'domain'], 'web domain');
9
10 const flags = args.parse(process.argv);
11
12 const FS_UNIAPP_API = process.env.FS_UNIAPP_API || flags.apiUrl;
13 const FS_PEP_WEB = process.env.FS_PEP_WEB || flags.pepwebUrl;
14 const FS_EMIS_WEB = process.env.FS_EMIS_WEB || flags.emiswebUrl;
15 const FS_CAMUNDA_HOST = process.env.CAMUNDA_HOST || flags.camundaHost;
16 const FS_CAMUNDA_ROOT = process.env.CAMUNDA_ROOT || flags.camundaRoot;
17 const FS_PEP_DOMAIN = process.env.FS_PEP_DOMAIN || flags.domain;
18
19 if (!FS_UNIAPP_API || !FS_PEP_WEB || !FS_PEP_DOMAIN) {
20   console.log('缺少启动参数, 异常退出');
21   args.showHelp();
22   process.exit(-1);
23 }

```

未使用代码

以下代码中第一个 `useEffect` 无用。

```

1 // client/src/sections/auth/containers/login.js
2
3   useEffect(() => {
4
5   }, [])
6

```

```

7   useEffect(() => {
8     if (error) {
9       message.error(error);
10      setPassword('')
11    }
12  }, [error])
13
14  useEffect(() => {
15    if (user && user.authorized) {
16      dispatch(push('/my-form'));
17    } else {
18      return RouteRequest.get(RouteTable.getPepwebUrl).then(res => {
19        if (res.url) {
20          window.location.href = `${res.url}/signin`;
21        }
22      });
23    }
24  }, [user])

```

以下代码中，`sections/auth` 模块未被使用，可去掉。

```

1  // client/src/app.js
2
3  import Auth from './sections/auth';
4
5  const App = props => {
6    ...
7    return (
8      <Layout
9        title={projectName}
10       sections={[Auth,
11                MyForm,
12                OaLedger
13              ]}
14      />
15    )
16  }

```

避免 `console.log`

生产环境下，不用 `console.log`，如 `client/src/sections/myform/actions/index.js` 中的代码。

```

1  console.log("actions:", actions);

```