项企平台代码走查

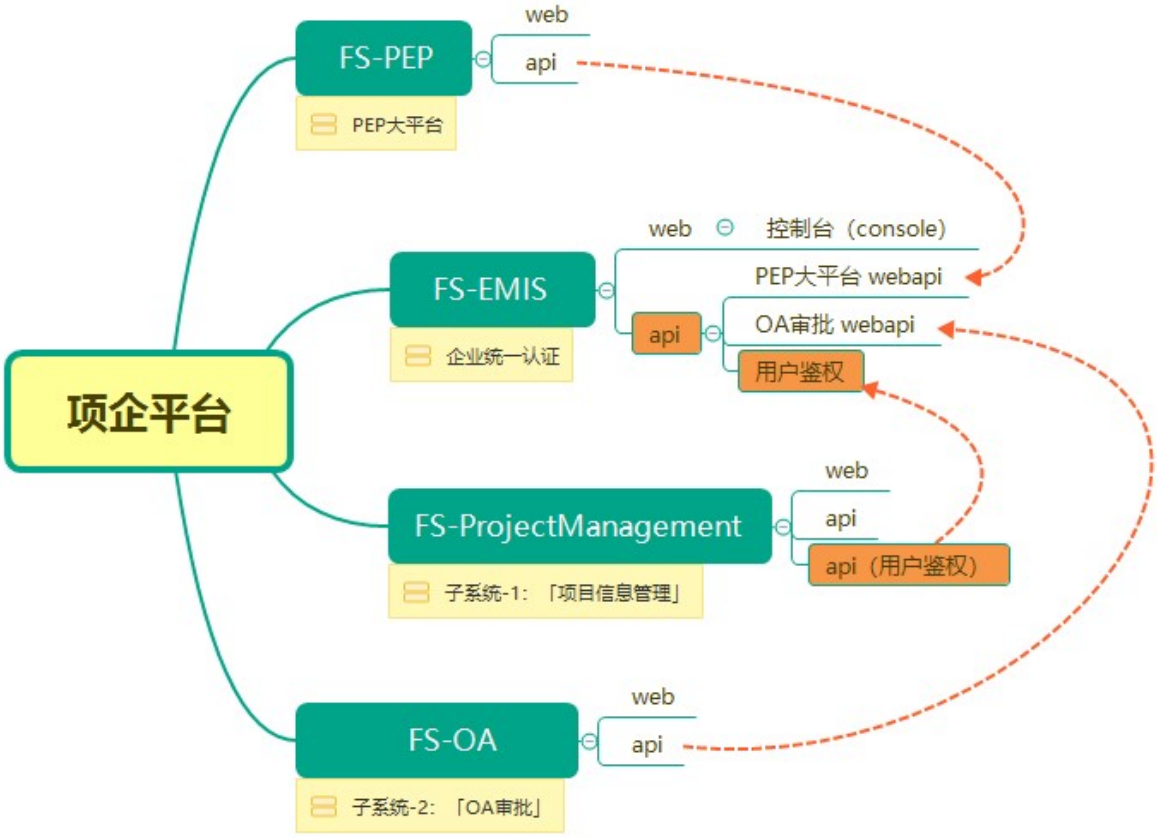
REVIEER: 彭玲 TIME: 2021/11/30

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

```
FS-EMIS\trunk\codes
├── api
└── 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**。

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

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`。

```
// app/lib/index.js
const apiLog = require('./middlewares/api-log');
const kafka = require('./middlewares/kafka');
module.exports.entry = function (app, router, opts) {
    // app.use(kafka(app, opts));
    // router.use(apiLog(app, opts));
}
```

中间件

是否真的需要作为中间件，如`business-rest.js`等。

```
// app/lib/middlewares/business-rest.js

'use strict';

const request = require('superagent');
const buildUrl = (url,token) => {
    let connector = url.indexOf('?') === -1 ? '?' : '&';
    return `${url}${connector}token=${token}`;
};

 function factory(app, router, opts) {
     return async function (ctx, next) {

         const token = ctx.fs.api.token;

```

```
        //console.log(username,password)
        const req = {
            get: (url, query) => {
                return request
                    .get(buildUrl(url,token))
                    .query(query)
            },
            post: (url, data, query) => {
                return request
                    .post(buildUrl(url,token))
                    .query(query)
                    //.set('Content-Type', 'application/json')
                    .send(data);
            },

            put: (url, data) => {
                return request
                    .put(buildUrl(url,token))
                    //.set('Content-Type', 'application/json')
                    .send(data);
            },

            delete: (url) => {
                return request
                    .del(buildUrl(url,token))
            },
        };

        app.business = app.business || {};
        app.business.request = req;

        await next();
    };
 }

module.exports = factory;
```

改为:

```
'use strict';

const request = require('superagent');

const buildUrl = (url,token) => {
    let connector = url.indexOf('?') === -1 ? '?' : '&';
    return `${url}${connector}token=${token}`;
};

function factory(app, router, opts) {
     let token = null;

     const req = {
         get: (url, query) => {
             return request
                 .get(buildUrl(url,token))
                 .query(query)
         },
```

```
        post: (url, data, query) => {
            return request
                .post(buildUrl(url,token))
                .query(query)
                .send(data);
        },

        put: (url, data) => {
            return request
                .put(buildUrl(url,token))
                .send(data);
        },

        delete: (url) => {
            return request
                .del(buildUrl(url,token))
        },
    };

    app.business = app.business || {};
    app.business.request = req;

    return async function (ctx, next) {
        token = ctx.fs.api.token;
        await next();
    };
}

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。

```
const userRes = await models.User.findOne({
    where: {
        account: params.username,
        password: password,
        delete: false,
    },
    attributes: { exclude: ['password'] },
});

let where = {
    expired: {
        $between: [
            moment().format('YYYY-MM-DD HH:mm:ss'),
            moment().add(opts.expiration, 'minutes').format('YYYY-MM-DD HH:mm:ss')
        ]
    }
}
let findObj = {
    where,
    attributes: ['token', 'expired'],
}
```

```
const userRes_ = await models.UserToken.findAll(
    findObj
);

if (!userRes) {
    ctx.status = 400;
    ctx.body = {
        "message": "账号或密码错误"
    }
} else if (!userRes.state) {
    ctx.status = 400;
    ctx.body = {
        "message": "该用户已被禁用"
    }
} else if (userRes.lock) {
    ctx.status = 400;
    ctx.body = {
        "message": "该用户已被锁定"
    }
} else if (userRes_.length > opts.maxLogNum) { // 这里查询数据库表 UserToken
    ctx.status = 400;
    ctx.body = {
        "message": `登录人数限制在${opts.maxLogNum}`
    }
} else {
    ...
}
```

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。

```
// app/lib/middlewares/authenticator.js

let authorizeToken = async function (ctx, token, opts) {
    const axyRes = await ctx.fs.dc.models.UserToken.findOne({
        where: {
            token: token,
            expired: { $gte: moment().subtract(opts.expiration, 'minutes').format('YYYY-MM-DD HH:mm:ss') }
        }
    });
    const { userInfo, expired } = axyRes;
    if (!expired || moment().subtract(opts.expiration, 'minutes').valueOf() <= moment(expired).valueOf()) { // 判断条件恒成立
        await ctx.fs.dc.models.UserToken.update({
            expired: moment(expired).add(opts.expiration, 'minutes').format()
        }, {
            where: {
                token: token,
            }
        });
```

```
        rslt = {
            'authorized': userInfo.authorized,
            'resources': (userInfo || {}).resources || [],
        };
        ...
}
```

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。

```
ctx.fs.logger.info(`[/login] START Time---: ${t0.format('YYYY-MM-DD HH:mm:ss.SSS')}`);

...

ctx.fs.logger.info(`[/login] END Time---: ${t11.format('YYYY-MM-DD HH:mm:ss.SSS')}--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()`。

```
// app/lib/controllers/auth/index.js
function getRedisInfo(key, ctx) {
    return new Promise((resolve, reject) => {
        if (ctx.redis) {
            ctx.redis.get(key, function (err, result) {
                if (err) {
                    console.log("请求redis失败:  ", err); // 控制台日志
                    return null;
                }
                resolve(result);
            })
        } else {
            resolve();
        }
    })
}
```

2. web 侧

启动参数

对启动参数赋初值，而非`空值`判断。

```
// // 启动参数
args.option(['p', 'port'], '启动端口');
args.option(['u', 'api-url'], 'webapi的URL');
args.option(['c', 'camundaHost'], 'camunda rest host');
args.option(['r', 'camundaRoot'], 'camunda rest root');
args.option('qnak', 'qiniuAccessKey');
args.option('qnsk', 'qiniuSecretKey');
args.option('qnbkt', 'qiniuBucket');
```

```
args.option('qndmn', 'qiniuDomain');
args.option(['d', 'domain'], 'web domain');
args.option('pepWebUrl', '大平台web的URL');
args.option('emisWebUrl', '统一认证web的URL');

const flags = args.parse(process.argv);

const FS_UNIAPP_API = process.env.FS_UNIAPP_API || flags.apiUrl;

const FS_CAMUNDA_HOST = process.env.CAMUNDA_HOST || flags.camundaHost;
const FS_CAMUNDA_ROOT = process.env.CAMUNDA_ROOT || flags.camundaRoot;
const ANXINCLOUD_QINIU_ACCESSKEY = process.env.ANXINCLOUD_QINIU_ACCESSKEY || flags.qnak;
const ANXINCLOUD_QINIU_SECRETKEY = process.env.ANXINCLOUD_QINIU_SECRETKEY || flags.qnsk;
const ANXINCLOUD_QINIU_BUCKET_RESOURCE = process.env.ANXINCLOUD_QINIU_BUCKET_RESOURCE || flags.qnbkt;
const ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE = process.env.ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE || flags.qndmn;
const FS_PEP_DOMAIN = process.env.FS_PEP_DOMAIN || flags.domain;
const FS_PEP_WEB_URL = process.env.FS_PEP_WEB_URL || flags.pepWebUrl;
const FS_EMIS_WEB_URL = process.env.FS_EMIS_WEB_URL || 'http://localhost:5000';

if (!FS_UNIAPP_API || !FS_CAMUNDA_HOST || !FS_CAMUNDA_ROOT || !ANXINCLOUD_QINIU_ACCESSKEY || !ANXINCLOUD_QINIU_SECRETKEY || !ANXINCLOUD_QINIU_BUCKET_RESOURCE || !ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE
    || !FS_PEP_DOMAIN
) {
    console.log('缺少启动参数，异常退出');
    args.showHelp();
    process.exit(-1);
}
```

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。

```
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？

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。

```
// client/src/sections/auth/containers/login.js

    useEffect(() => {

    }, [])

    useEffect(() => {
        if (error) {
            message.error(error);
            setPassword('')
        }
    }, [error])
```

无用的注释

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

```
        // PEP首页的“待办事项”、“最新单位公告”定时
        // homePageInterval = setInterval(() => {
        //     if (userId) {
        //         dispatch(getCompanyNotice(userId)).then(res => {
        //             setauditUnit(res.payload.data.length)
        //             setShowCompanyNotice(res.payload.data);
        //         })
        //     }
        //     dispatch(getMyAuditList(params)).then(res => {
        //         setauditList(res.payload.data.slice(0, 5))
        //         setauditCunt(res.payload.data.length)
        //         setHomeData(false)
        //     })
        // }, 60000)
```

命名规范性

函数名

例如，auth 模块 getIsLogout() 命名。

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

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。

```
    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`。

```
const jumpSystem = (v, code) => {
   ...
    window.open(v.url);
   ...
}
// 调用
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?

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

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

```
FS-ProjectManagement\trunk
├── web
└── 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。

```
// 启动参数
args.option(['p', 'port'], '启动端口');
args.option(['u', 'api-url'], 'webapi的URL');
args.option(['t', 'ty-api-url'], '统一管理平台webapi的URL');
args.option(['f', 'ty-web-url'], '统一管理平台web的URL');
args.option('pepWebUrl', '项企大平台web的URL');
args.option('pmApiUrl', '项目管理api的URL');    //商用环境用域名时配置该参数
args.option('qnak', 'qiniuAccessKey');
args.option('qnsk', 'qiniuSecretKey');
args.option('qnbkt', 'qiniuBucket');
args.option('qndmn', 'qiniuDomain');
args.option(['d', 'domain'], 'web domain');

const flags = args.parse(process.argv);
const FS_PROJECTMGT_API = process.env.FS_PROJECTMGT_API || flags.apiUrl;
const FS_TYMG_API = process.env.FS_TYMG_API || flags.tyApiUrl;
const FS_TYMG_WEB = process.env.FS_TYMG_WEB || flags.tyWebUrl;
const FS_PEP_WEB = process.env.FS_PEP_WEB || flags.pepWebUrl;
const FS_PM_API = process.env.FS_PM_API || flags.pmApiUrl;
const ANXINCLOUD_QINIU_ACCESSKEY = process.env.ANXINCLOUD_QINIU_ACCESSKEY ||
flags.qnak;
const ANXINCLOUD_QINIU_SECRETKEY = process.env.ANXINCLOUD_QINIU_SECRETKEY ||
flags.qnsk;
const ANXINCLOUD_QINIU_BUCKET_RESOURCE =
process.env.ANXINCLOUD_QINIU_BUCKET_RESOURCE || flags.qnbkt;
const ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE =
process.env.ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE || flags.qndmn;
const FS_PEP_DOMAIN = process.env.FS_PEP_DOMAIN || flags.domain;

if (!FS_PROJECTMGT_API || !FS_TYMG_API || !FS_TYMG_WEB || !FS_PEP_WEB ||
!ANXINCLOUD_QINIU_ACCESSKEY || !ANXINCLOUD_QINIU_SECRETKEY ||
    !ANXINCLOUD_QINIU_BUCKET_RESOURCE ||
!ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE || !FS_PEP_DOMAIN) {
    console.log('缺少启动参数，异常退出');
    args.showHelp();
    process.exit(-1);
}
```

未使用代码

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

```
// client/src/app.js

let sections = [
    Auth,
    personalDetail,
    ProjectManage,
    ConstructionLog,
    ProjectApprove,
```

```
    DocumentCenter,
    OrganizationManage,
    clientManage,
    clientVisit,
    reserveProjectManage,
    ProjectAlternatives,
    // approvalCenter,
    //暂时隐藏
    quotationManage,
    quotationManageSales
];
```

console.log

代码中很多`console.log`及其注释。

```
// client/src/layout/containers/layout/index.js
itemRender = (route, params, routes, paths) => {
    // console.log(route, params, routes, paths)
    ...
    console.log(paths_)
}
```

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。

```
// // 启动参数
args.option(['p', 'port'], '启动端口');
args.option(['u', 'api-url'], 'webapi的URL');
args.option(['c', 'camundaHost'], 'camunda rest host');
args.option(['r', 'camundaRoot'], 'camunda rest root');
args.option('pepWebUrl', '项企大平台web的URL');
args.option('emisWebUrl', '企业统一认证管理平台web的URL');
args.option(['d', 'domain'], 'web domain');

const flags = args.parse(process.argv);

const FS_UNIAPP_API = process.env.FS_UNIAPP_API || flags.apiUrl;
const FS_PEP_WEB = process.env.FS_PEP_WEB || flags.pepWebUrl;
const FS_EMIS_WEB = process.env.FS_EMIS_WEB || flags.emisWebUrl;
const FS_CAMUNDA_HOST = process.env.CAMUNDA_HOST || flags.camundaHost;
const FS_CAMUNDA_ROOT = process.env.CAMUNDA_ROOT || flags.camundaRoot;
const FS_PEP_DOMAIN = process.env.FS_PEP_DOMAIN || flags.domain;

if (!FS_UNIAPP_API || !FS_PEP_WEB || !FS_PEP_DOMAIN) {
    console.log('缺少启动参数，异常退出');
    args.showHelp();
    process.exit(-1);
}
```

未使用代码

以下代码中第一个`useEffect`无用。

```
// client/src/sections/auth/containers/login.js

    useEffect(() => {

    }, [])

    useEffect(() => {
        if (error) {
            message.error(error);
            setPassword('')
        }
    }, [error])

    useEffect(() => {
        if (user && user.authorized) {
            dispatch(push('/my-form'));
        } else {
            return RouteRequest.get(RouteTable.getPepWebUrl).then(res => {
                if (res.url) {
                    window.location.href = `${res.url}/signin`;
                }
            });
        }
    }, [user])
```

以下代码中，`sections/auth`模块未被使用，可去掉。

```
// client/src/app.js

import Auth from './sections/auth';

const App = props => {
    ...
    return (
        <Layout
            title={projectName}
            sections={[Auth,
                MyForm,
                OaLedger
            ]}
        />
    )
}
```

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。

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