灰度发布：前端预研

AUTHOR: 彭玲 TIME: 2021/6/30

预研成果

1. web 侧

通过 Cookie 控制 `灰度版本` 与 `生产版本`。

实现思想

- 通过 `/login` 登录接口查询用户使用的服务版本，设置灰度发布用 cookie（值定义为 `always` 和 `never`）
  - `always`：使用 `灰度版本`
  - `never`：使用 `生产版本`
- 脚手架中设置 cookie 默认值

登录接口

灰度发布用 cookie 设置：

- `cookie-gray-released` 有效期：每次登录后顺延 60 天
- 测试案例简化：超级管理员（`id`=1）设置 `always` 灰度版本，其他用户设置 `never` 生产版本

```
// app/lib/controllers/auth.js

function login() {
    const SUPER_ADMIN_ID = 1;
    const DAY_60_IN_MILLI = 5184000000;
    let gr = userInfo.id == SUPER_ADMIN_ID ? 'always' : 'never'; // userInfo.id
= user.id
    ctx.cookies.set('cookie-gray-released', gr, { maxAge: DAY_60_IN_MILLI });
}
```

脚手架

灰度发布用 cookie 默认值设置（默认值`never`）：

```
// node_modules/fs-web-server-scaffold

function errorHandler() {
    return function* (ctx, next) {
        try {
            let cgr = ctx.cookies.get('cookie-gray-released')
            if (!cgr) {
                ctx.cookies.set('cookie-gray-released', 'never')
            }

            yield next();
        } catch (err) {
            app.fs.logger.log("error", "[FS-ERRHD]", err);
            //simple process.
            //@Todo 500 page; 400...
            ctx.status = 500;
            ctx.body = 'internal server error';
        }
    };
}
```

2. web-api 侧

通过`http 请求头`控制`灰度版本`与`生产版本`。

实现思想

- 通过`/login`登录接口查询用户使用的服务版本（`always`和`never`），使用全局对象`app`缓存该版本信息（使用`token`标识登录用户）
- 脚手架中根据请求参数`token`获取`app`缓存的服务版本，设置灰度发布用`请求头`

登录接口

- 路由器中注入`app`全局对象

```
// app/lib/routes/auth/index.js

app.fs.api.logAttr['POST/login'] = { content: '登录', visible: true };
router.post('/login', auth.login(app));
```

- 控制器

灰度发布用`app.fs.canary`属性设置：

```
// app/lib/controllers/auth.js

function login(app) {
    const token = uuid.v4();

    let gr = userInfo.id == SUPER_ADMIN_ID ? 'always' : 'never';
    app.fs.canary = app.fs.canary || {};
    app.fs.canary[token] = gr;
}
```

脚手架

灰度发布用`请求头`设置（默认值`never`）：

```
// node_modules/fs-web-server-scaffold

function errorHandler() {
    return function* (ctx, next) {
        try {
            const token = ctx.query.token
            let gr = token && app.fs.canary && app.fs.canary[token]
            ctx.request.header = Object.assign({}, ctx.header, { 'gray-
released': gr || 'never' })

            yield next();
        } catch (err) {
            app.fs.logger.log("error", "[FS-ERRHD]", err);
            //simple process.
            //@Todo 500 page; 400...
            ctx.status = 500;
            ctx.body = 'internal server error';
        }
    };
}
```

开发指导

数据库设计

用户表扩展属性`canary`，用于标识用户使用的服务版本：`灰度版本` or `生产版本`

- `varchar(255)`类型：统一 web 与 api 的版本，即全用`always`灰度版本或全用`never`生产版本
- `jsonb`类型：分别记录各服务（web-api、web-console、web-project 等）的版本

Redis 解决 openapi 服务

- redis 缓存管理：`config_center` 进程，维护 `t_user_token` 中**未过期的**记录
  - redis `key`：`t_user_token` 的 `token` 值
  - redis `value`：`{ userId, canary }`（从 `t_user_token` 的 `user_info` 中提取）
- redis 缓存使用：web-api 服务

  脚手架中优先从登录接口获取 `app.fs.canary[token]`，其次从 redis 读取：

```
// node_modules/fs-web-server-scaffold

const token = ctx.query.token
const grFromLogin = token && app.fs.canary && app.fs.canary[token]
const grFromRedis = await redis.get(token).canary
ctx.request.header = Object.assign({}, ctx.header, { 'gray-released':
grFromLogin || grFromRedis || 'never' })
```

数据库 CUD 维护

- 管理系统
- 数据库脚本