YOG2框架介绍

何方石@FEX

首先…

- why YOG2
  - 快速开始
  - 结合FIS
  - 模块拆分
  - 功能封装
  - 统一升级

YOG2框架简介

- 基于Express
- FIS静态资源管理
- 模块拆分
- 路由系统
- 后端服务管理
- 插件系统

YOG2框架简介

业务框架

静态资源管理

yog-view

yog-swig

yog-bigpipe

内部服务

passport

wise

uuap

自定义插件

yog-ral

yog-log

基础框架

Plugins

Dispatcher

Config

运行时

Node

Daemon

Monitor

脚手架

FIS静态资源管理

- 基于swig后端模板引擎
  - {%require%}
  - {%widget%}
- mod.js
  - require
  - require.async

FIS静态资源管理

- bigpipe
  - 类似lazyrender
  - 占位+JS渲染
  - 静态资源按需加载
  - 不同之处
    - chunk输出
    - 无法gizp

```
// /home/client/page/index.tpl

{%
    widget "spa:widget/imageList/imageList.tpl"
    id = "bookList"
    mode = "async"
%}
```

```
// /home/server/action/index.js

module.exports = function (req, res, next) {
    req.bigpipe.bind('bookList', yog.ral('WENKU', {
        path: 'api/book',
        query: req.query
    }));
    yog.ral('WENKU', {
        path: 'api/userinfo'
    }).then(res.curryRender('index.tpl')).catch(next);
}
```

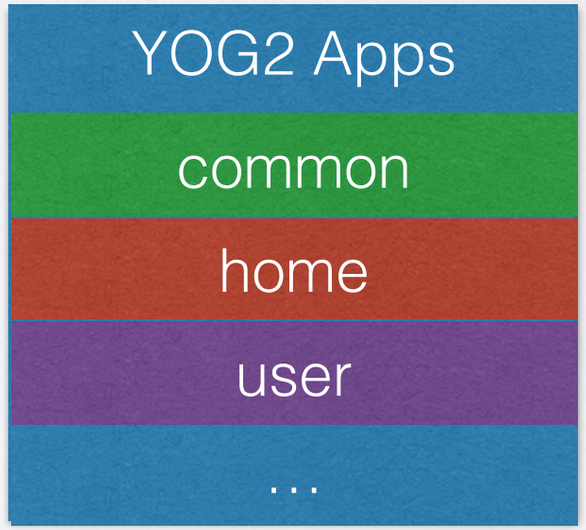
FIS静态资源管理

- pagelet
  - 异步请求后端渲染
  - 占位+JS渲染
  - 静态资源按需加载

```
// /home/client/page/index.tpl

{%
    widget "spa:widget/imageList/imageList.tpl"
    id = "bookList"
    mode = "quickling"
%}

// /home/server/action/index.js

module.exports = function (req, res, next) {
    req.bigpipe.bind('bookList', yog.ral('WENKU', {
        path: 'api/book',
        query: req.query
    }));
    if (req.bigpipe.pagelets.length !== 0) {
        res.render('index.tpl', {});
    } else {
        yog.ral('WENKU', {
            path: 'api/userinfo'
        }).then(res.curryRender('index.tpl')).catch(next);
    }
}

// /home/client/static/index.js

BigPipe.load('bookList');
```

YOG2子项目拆分

- 目标
  - 独立编译、独立上线
  - 独立的路由控制
  - 跨模块调用能力



YOG2模块拆分

- YOG2 App

```
├─client                    # 前端代码
│   ├─page                  # 前端页面
│   ├─static                # 前端非模块化静态资源
│   │   ├─css
│   │   └─js
│   └─widget                # 前端组件
├─fis-conf.js               # FIS编译配置
└─server                    # 后端代码
    ├─action                # Action是指MVC中的路由动作，处理页面请求
    ├─lib                   # 可以存放一些通用库
    ├─model                 # 可以存放一些数据层代码，如后端API请求等
    └─router.js             # AppRouter路由，用于处理自动路由无法满足的需求
```

YOG2模块拆分

- YOG2 Project

```
├─yog
├─app                // 模块服务端代码
├─bin
├─conf               // 配置文件
│   ├─plugins
│   └─ral
├─plugins            // 插件目录
├─static             // 静态资源
└─views              // 后端模板
```

YOG2模块拆分

YOG2模块拆分

- YOG2 Release

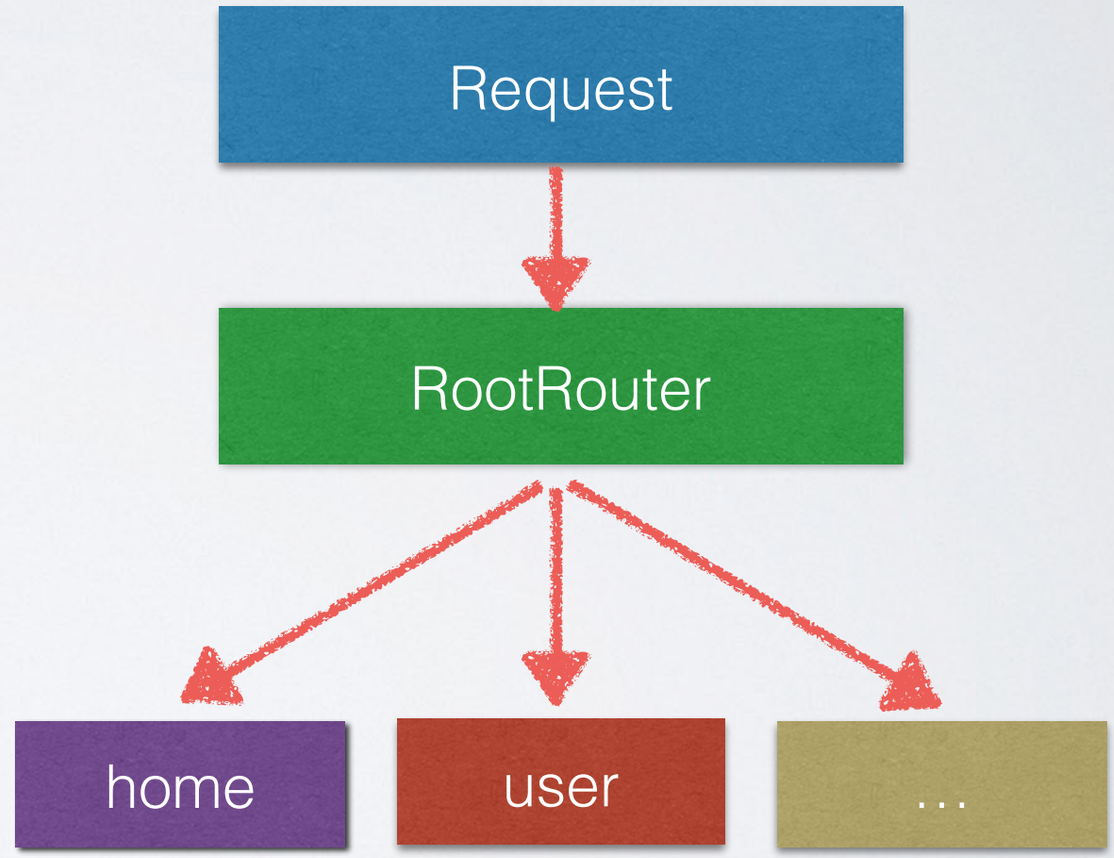
```
fis.config.set('roadmap.path', [{
    reg: /^\/client\/widget\/(.+\.tpl)$/i,
    isMod: true,
    id: 'widget/$1',
    url: '${namespace}/widget/$1',
    release: '${template}/${namespace}/widget/$1'
}, {
    reg: /^\/client\/(.*)/i,
    id: '$1',
    release: '${static}/${namespace}/$1'
}, {
    reg: /^\/server\/(.+)/i,
    useMap: false,
    useCompile: false,
    useHash: false,
    useDomain: false,
    release: '${app}/${namespace}/$1'
}]);
```

YOG2模块拆分

- 跨模块调用
  - 前端
    - {%widget ‘common:widget/nav/nav.tpl’ %}
    - require(‘common:widget/constant/constant.js’);
- 后端
  - yog.require(‘common/libs/util.js’);

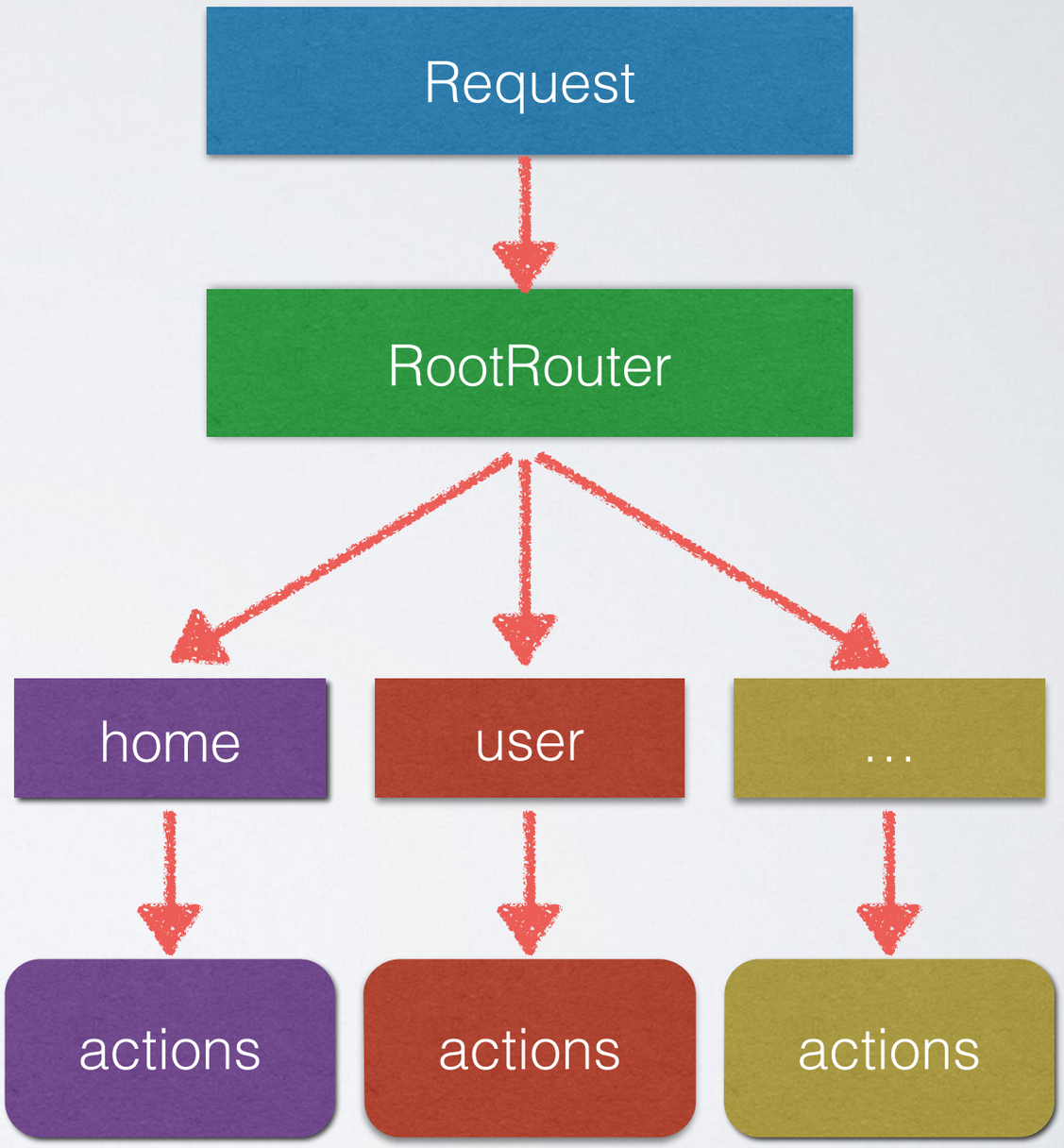
YOG2路由系统

- 基于Express.Router
- 提升易用度
- 模块拆分友好
- 自动寻找 vs 主动注册
- 灵活扩展



YOG2路由系统

- 模块拆分友好
  - RootRouter决定App转发
  - AppRouter决定Action转发



YOG2路由系统

- 自动寻找 vs 主动注册
  - Express

```
app.get('/', function (req, res) {
    res.send('hello world');
});
```

  - YOG2

```
// home/server/action/index.js
module.exports = function (req, res) {
    res.send('hello world');
};
```

YOG2路由系统

- 自动寻找规则

http://www.example.com/home/index

=>

app/home/action/index.js

http://www.example.com/home/doc/detail

=>

app/home/action/doc/detail.js

http://www.example.com/home/doc/detail

=>

app/home/action/doc/detail/index.js

YOG2路由系统

- e.g. RootRouter自定义路由

```
// /conf/plugins/dispatcher.js

module.exports = function (router) {
    router.use('/custom', yog.dispatcher.router('home'));
    router.get('/somespecial', yog.dispatcher.action('home/doc/detail'));
};
```

- e.g. AppRouter自定义路由

```
// /home/server/router.js

module.exports = function (router) {
    router.get('/search/:keyword', router.action('search'));
    router.delete('/comment/:id', router.action('comment').delete);
};
```

YOG2路由系统

- 灵活扩展
  - Everything in Express.Router
  - Router Middleware
  - Action Middleware

YOG2路由系统

- e.g. RootRouter 扩展RAL自动注入

```
module.exports = function (router) {
    router.use(function (req, res, next) {
        req.ral = function (name, options) {
            // inject query
            options.query = options.query || {};
            var userAgent = req.headers['user-agent'];
            if (/iPhone|iPad|Mac/.test(userAgent)) {
                options.query.p = 2;
            } else if (/Android/.test(userAgent)) {
                options.query.p = 3;
            } else if (/Windows Phone/.test(userAgent)) {
                options.query.p = 4;
            } else {
                options.query.p = 1;
            }
            // inject headers
            options.headers = _.extend(options.headers || {}, {
                "x-bd-logid": yog.log.getLogID(),
                "Cookie": "BDUSS=" + (req.cookies.BDUSS || req.query.bduss)
            });
            return yog.ral(name, options);
        };
    });
};
```

YOG2路由系统

- e.g. AppRouter扩展 res.api

```
// /home/server/router.js

module.exports = function (router) {
    router.use(injectAPIRes);
};

function injectAPIRes(req, res, next) {
    res.api = function (err, data) {
        if (err && err.errno) {
            if (!err.errno) {
                return next(err);
            }
            res.json({
                errno: err.errno,
                errmsg: err.message
            });
        } else {
            res.json({
                errno: 0,
                data: data
            });
        }
    };
    next();
}
```

YOG2路由系统

- e.g. AppRouter扩展Passport验证

```
// /home/server/router.js

module.exports = function (router) {
    router.use(getSessionInfo);
};

function getSessionInfo(req, res, next) {
    yog.passport.getUser(req.cookies.bduss, function (err, data) {
        if (err) {
            return next(err);
        }
        req.user = data;
        next();
    });
}
```

YOG2路由系统

- e.g. Action添加异步授权验证

```
// /home/server/action/index.js

module.exports = function (req, res, next) {
    checkAuth(req.user, auth.ADMIN, function (err, data) {
        if (err) return next(err);
        res.send('hi');
    });
};
```



```
// /home/server/action/index.js

module.exports = auth(auth.ADMIN, function (req, res, next) {
    res.send('hi');
});

function auth(role, func) {
    return function (req, res, next) {
        checkAuth(req.user, role, function (err, data) {
            if (err) return next(err);
            func(req, res, next);
        });
    };
}
```

YOG2后端服务管理理

- 解决的问题
  - 后端服务配置统一管理
  - 封装异常处理、超时重试，提升系统稳定性
  - 封装日志，便于线上问题追查
  - 抽象请求协议、数据格式与数据编码，统一用户接口

YOG2后端服务管理

- 后端服务配置统一管理
- 配置 > 硬编码
- 开发环境与生产环境管理
- 配置与业务解耦

```
// /conf/ral/WENKU.js

module.exports = {
    'WENKU': {
        unpack: 'json',
        pack: 'querystring',
        method: 'GET',
        encoding: 'gbk',
        balance: 'random',
        protocol: 'http',
        retry: 2,
        timeout: 500,
        server: [{
            host: 'wenku.baidu.com',
            port: 80
        }]
    }
};

// /home/server/action/index.js
req.headers.host = null;
yog.ral('WENKU', {
    data: req.query,
    headers: req.headers,
    path: '/api/book'
})
.on('data', res.json.bind(res))
.on('error', next);
```

YOG2后端服务管理

- 异常处理
  - 多个阶段都需要异常处理
  - 除数据打包外，每个阶段的异常都需要重试
  - 都可以抽象与封装

YOG2后端服务管理

- 封装日志
  - 多个阶段都有日志
  - 记录每个阶段的耗时
  - 记录请求失败原因

[yog-ral] [cluster 6][RAL] request end service=hades requestID=862014278014 conv=form/json prot=http method=POST path=/api/getgroupbyserviceid remote=hades.baidu.com:80 cost=27.373 talk=27.259 write=26.723 read=0.285 pack=0.076 unpack=0.088 retry=0/1

[yog-ral] [cluster 20][RAL] request failed service=hades requestID=161480974229 conv=form/json prot=http method=POST path=/namingservice/getinstancebygroupid remote=hades.baidu.com:80 cost=1038.869 talk=1038.754 write=1038.635 read=0.000 pack=0.069 unpack=0.000 retry=0/0 errmsg=Server Status Error: 500

YOG2后端服务管理理

- 抽象请求协议、数据格式
  - 正交设计
  - 支持多种请求协议 http, https, soap, nshead
  - 支持多种数据格式 querystring, form, formdata, json, stream
  - 配置与业务解耦

YOG2插件系统

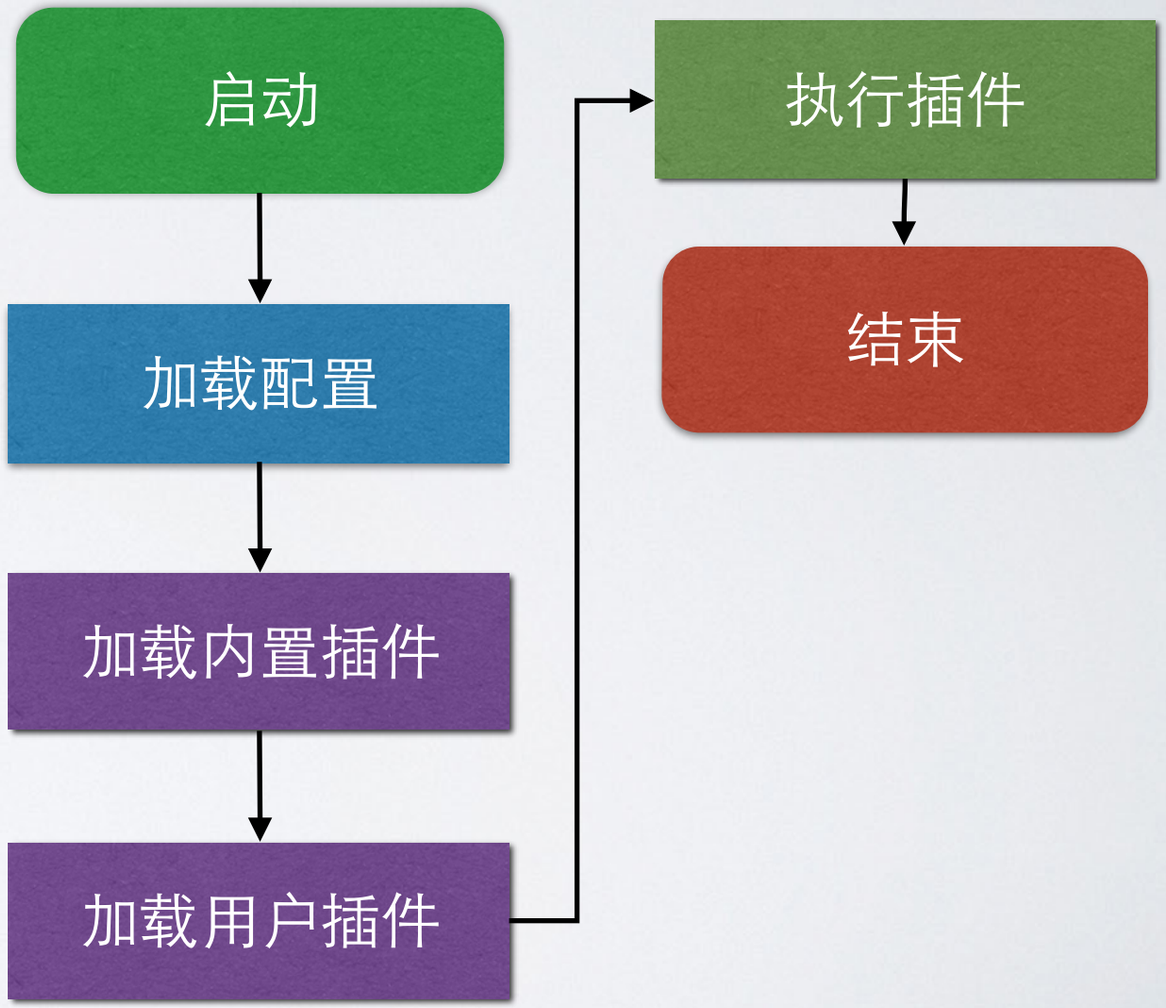
- 目标
  - 框架功能通过插件系统实现功能与配置的分离
  - 逻辑由插件自身实现
  - 配置由插件系统统一管理

YOG2插件系统

- 为什么要使用插件系统
  - YOG2为了方便使用，默认引入了多个中间件
    - 在app.js中引用会与用户代码混杂，无法升级
    - 在yog2-kernel中引用会导致用户黑盒严重，不够灵活
- 解决办法
  - 逻辑内置、配置暴露，由插件系统管理

YOG2插件系统

- 插件初始化顺序
- 用户插件可以覆盖内置插件
- 插件之间可以设置依赖关系
- 可以通过配置禁用指定插件



YOG2插件系统

- 内置插件
  - dispatcher 自动路由分发插件，提供全局函数yog.dispatcher
  - http 中间件管理插件，通过配置，用户可以方便的管理中间件加载顺序和新增中间件
  - log 日志插件，提供全局函数yog.log
  - ral 后端服务管理插件，提供全局函数yog.ral
  - views FIS静态资源管理与模板插件

YOG2插件系统

- 插件命名规范
  - 所有用户插件均在 /plugins目录中
  - 每个插件都有一个独立的文件夹
  - 文件夹中的 index.js 为插件的入口
  - index.js 中的 module.exports.pluginA 是插件的初始化逻辑
  - index.js 中的 module.exports.pluginA.defaultConf 是插件的默认配置

YOG2插件系统

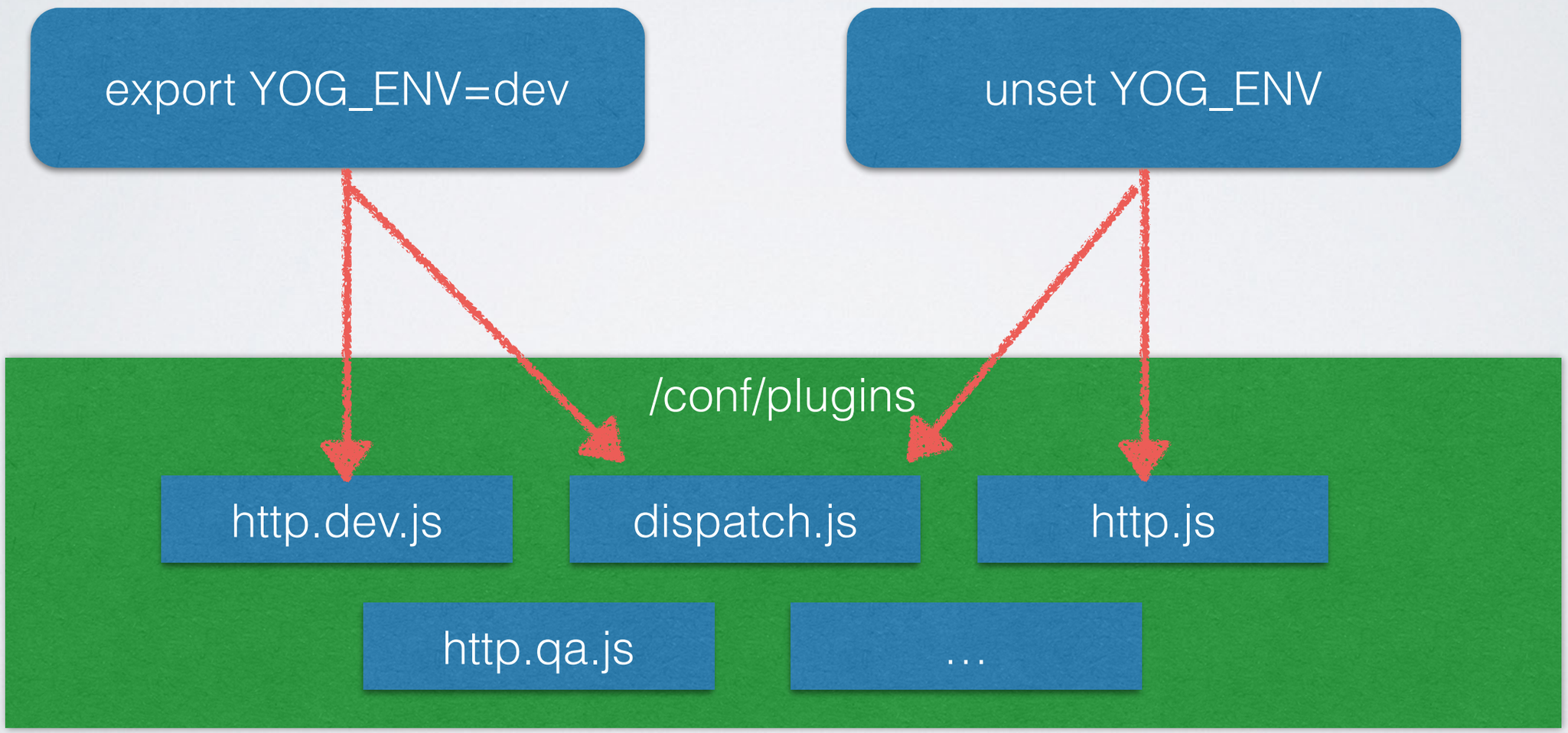
- 配置管理
  - 配置统一在 /conf/plugins 中管理
    - 目录中所有 js 与 json 文件均会被加载
    - 通过文件内容作为命名规范而非文件名
    - 通过环境变量区分开发环境与生产环境的配置

YOG2插件系统

- 配置命名规范
  - 与文件名无关
  - 一个文件中可以保存多个插件的配置
  - module.exports.pluginA = {} 表示pluginA会使用此配置进行初始化
  - 会根据文件后缀名与YOG_ENV环境变量加载配置

YOG2插件系统

- 开发与生产环境配置



YOG2插件系统

- 基于插件的中间件管理
  - 中间件使用插件封装
  - 中间件管理器本身也是插件 http插件
  - http插件可以按配置顺序加载其余中间件插件
  - http插件只负责调用中间件插件，如何加载中间件由中间件插件负责

YOG2插件系统

- 中间件插件
  - 返回一个函数
  - 函数中加载中间件

```
// plugins/log/index.js

var logger = require('yog-log');
var path = require('path');

module.exports.log = function (app, conf) {
    yog.__defineGetter__('log', function () {
        return logger.getLogger(conf);
    });
    return function () {
        app.use(logger(conf));
    };
};

module.exports.log.defaultConf = {
    'app': 'yog',
    'data_path': path.join(yog.ROOT_PATH, 'tmp'),
    'log_path': path.join(yog.ROOT_PATH, 'log'),
    'use_sub_dir': 1
};
```

YOG2插件系统

- e.g. 封装ral-failover-promise接口插件

```
var P = require('bluebird');

module.exports.ralPromise = ['ral',
    function (app, conf) {
        var ral = yog.ral;
        yog.ral = function (name, options) {
            var resolver = P.pending();
            // use promise interface
            ral(name, options).on('data', function (data) {
                resolver.resolve(data);
            }).on('error', function (error) {
                // only reject when failover is false
                if (options.failover !== false) {
                    resolver.resolve(options.failover || {});
                } else {
                    resolver.reject(error);
                }
            });
            return resolver.promise;
        };
    }
];
```

```
var P = require('bluebird');

var wenkuSearch = yog.ral('WENKU', {
    data: req.query,
    path: '/api/book',
    failover: false
});

var cpro = yog.ral('CPRO', {
    data: req.query,
    path: '/api/ad',
    failover: {
        count: 0
    }
});

P.props({
    bookList: wenkuSearch,
    cpro: cpro
})
.then(res.curryRender('index.tpl'))
.catch(next);
```

预告

- APP代码热加载，告别重启 (开发期)
- node-ral添加mock/stub
- 文档补充，添加更多DEMO
- 日志系统性能改进

THANKS

- http://fex.baidu.com
- http://agroup.baidu.com
- http://github.com/fex-team
- http://github.com/fex-team/yog2

FEX

Web前端研发部