FEDAY 北京 2017

第三届

FEDAY(前端开发日)

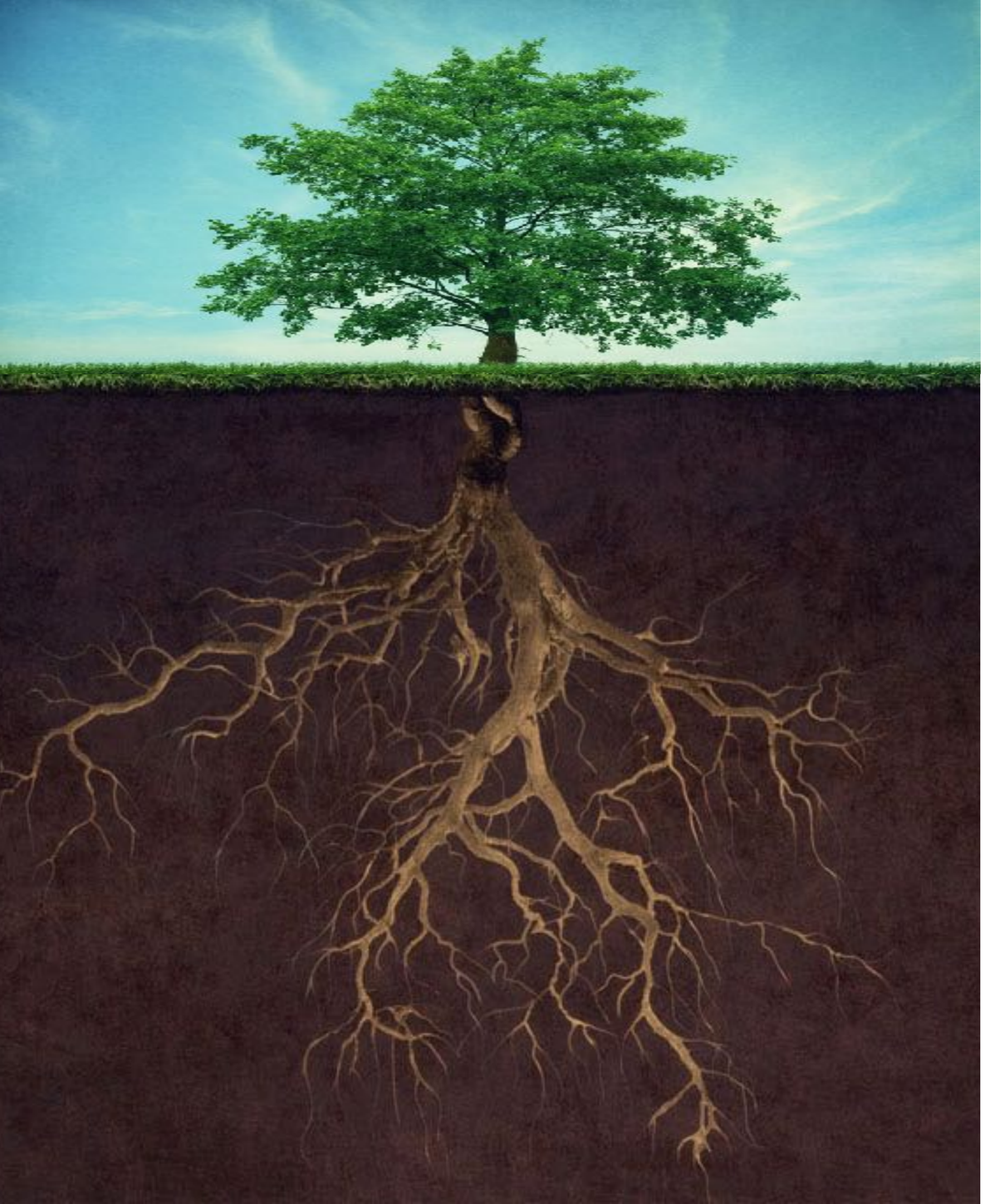
UCLOUD 100offer upyun Broadview 图灵教育 GenePlan StuQ 斯达克学院 CSDN 不止于代码 W3C plus 掘金 前端早读课 中生代技术 segmentfault 码云 Gitee.com

杨永林
链家网前端架构师

链家网前端工程实践

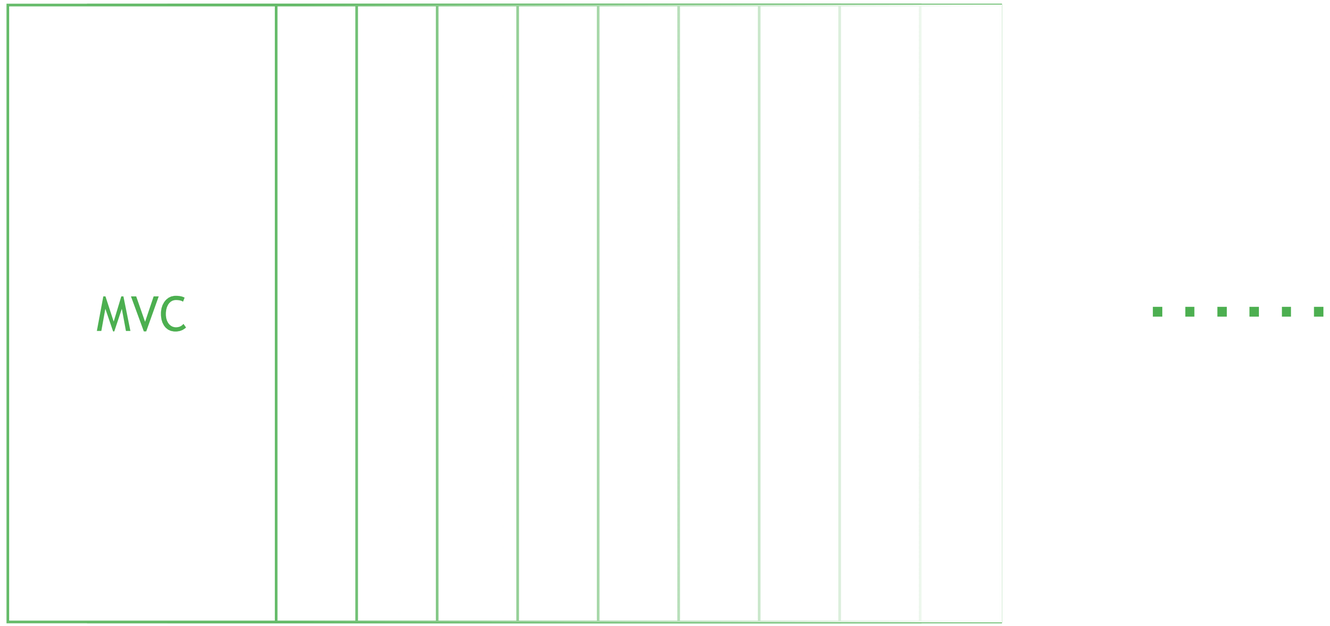
2014

通过互联网平台实现买方、卖方、经纪人的连接与互通，形成房产生态闭环。

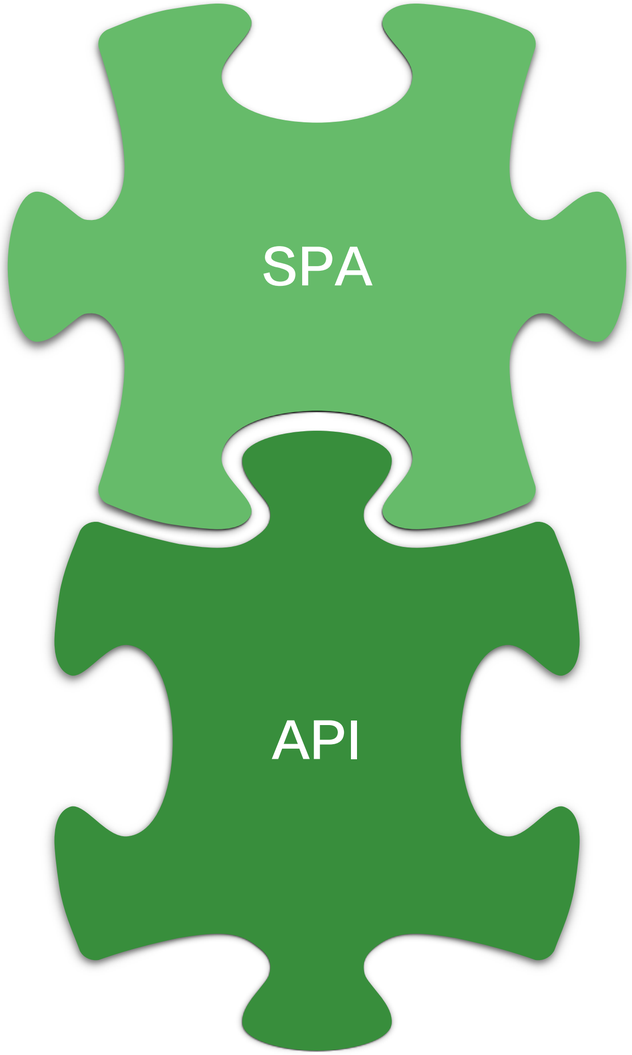


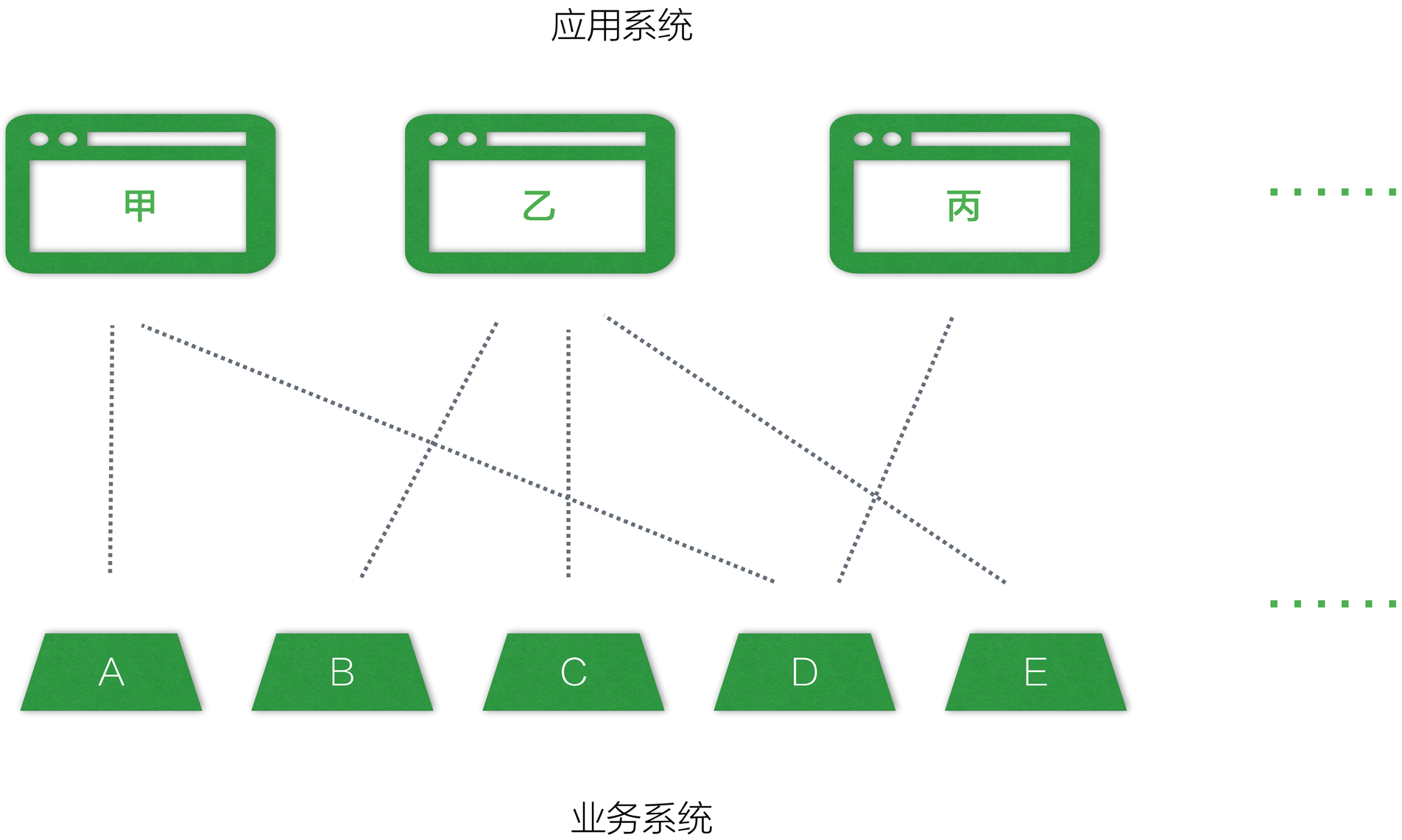
- 系统数量众多
- 生命周期长
- 开发节奏不稳定
- 访问量低
- 每一个环节都很重要

- 实现简难维护
- 查路题越本高
- 周边配发混乱
- 耦性好长



- 上线模块变多
- 跨系统调用问题突出
- 数据接口定义冲突严重
- 系统脆弱不便监控

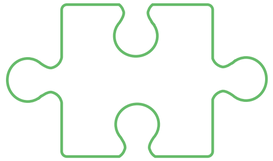



应用系统
甲
乙
丙
A
B
C
D
E
业务系统

- 多业务方
- 功能简单
- 开发节奏不确定
- 内容变化多端
- 经常要分享

“单一系统不灵活；

多个系统不好管。”



静态资源

controller

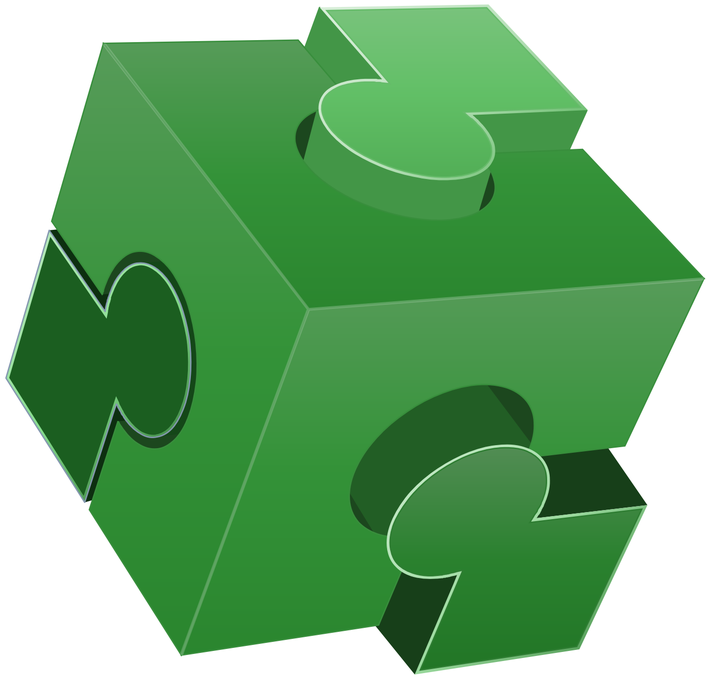
页面模板

配置信息

发布到静态池

在主服务中action目录建立目录，放置controller和页面模版

拷贝到配置信息目录



独立运行的主服务

运行时

接口封装

流程封装

应用API

...

- 培训成本高
- 环境部署麻烦
- 不好招人

实际业务工作不复杂

用Node做一个框架，我们需要做什么

通过搭建一个易于编写，便于开发部署与问题查找的稳定框架，高效地完成业务开发任务。

FIN

稳定

编写简单

便于查找问题

方便开发环境部署

以的合适框架为基础，降低开发成本。

koa

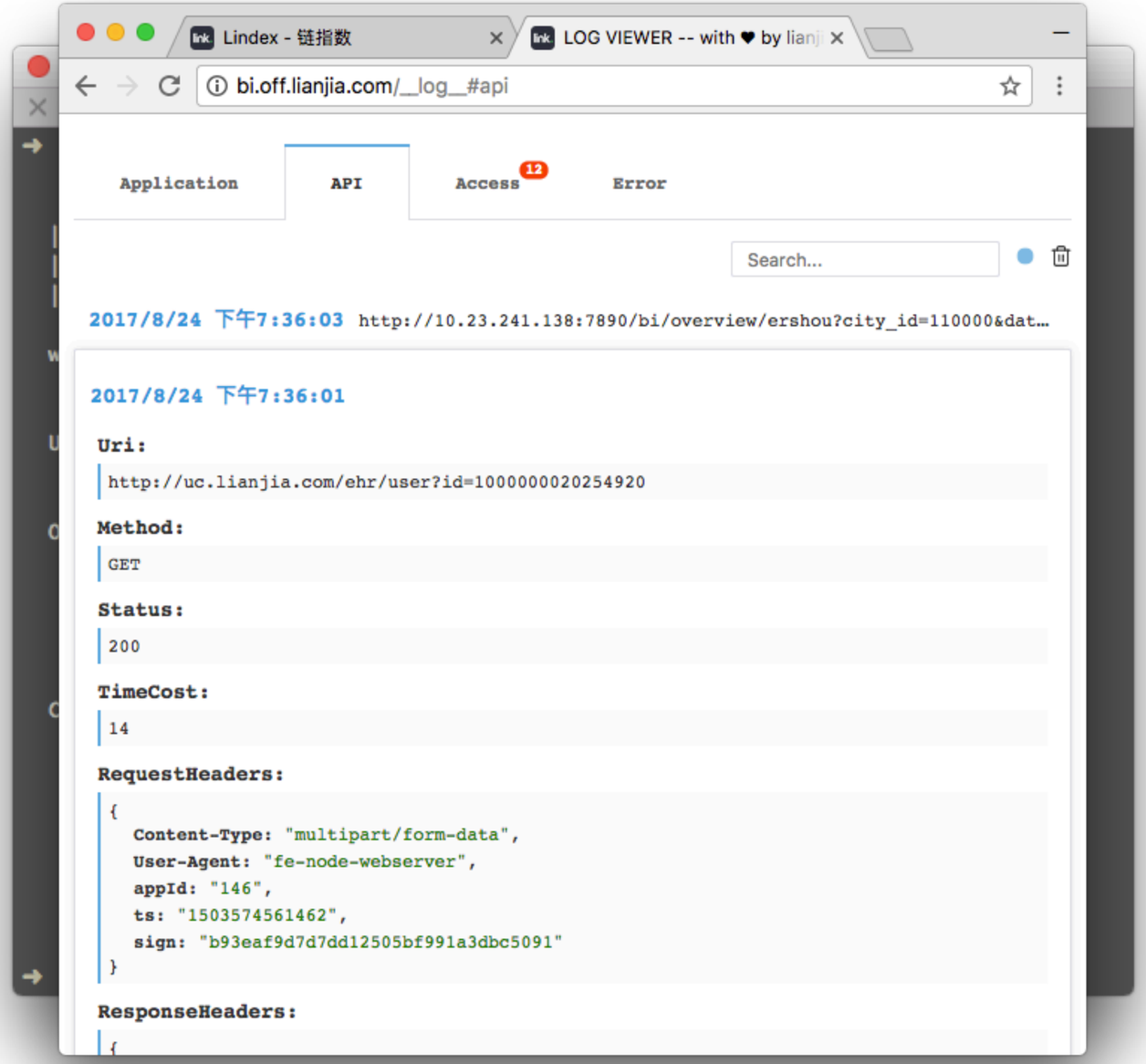
next generation web framework for node.js

模版

- configs
  - _prod.js
  - action.js
  - app.js
  - env.js
  - formparser.js
  - log.js
  - login.js
  - logviewer.js
  - mysql.js
  - redirect.js
  - redis.js
  - response.js
  - rewrite.js
  - socket.js
  - static.js
  - view.js

```
export default class EditorAjax {

  constructor () {

  }

  static async getCardData (key, type) {
    const {errno, errmsg, data} = await API.fangObj.toSearch({keyword
    const ids = []

    if (data.list) {
      for (let item of data.list) {
        if (item.type === type) ids.push(item.id)
      }

      if (ids.length) {
        return await API.editorajax.getData({cid: ids.join(',')})
      }
    }
    return {'errno' : 200, 'errmsg' : '', data : {'list' : []}}
  }

  static async searchEntity (key) {
    return await API.editorajax.searchentity({ key })
  }

  static async getImg (pid) {
    return await API.editorajax.searchimg({ pid })
  }


}
```

提供什么服务



Cli工具

既然可以让代码格式固化和模版化，Cli工具可以更方便的让工程师创建代码，然后填空，提高效率，减少错误。

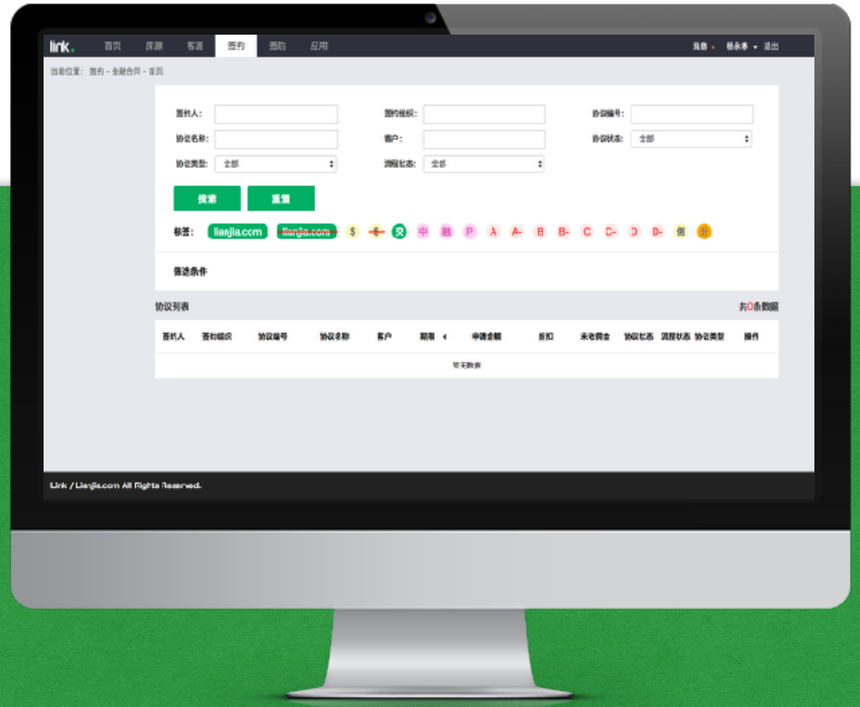
封装通用业务接口

封装了登录，权限校验，文件上传，S3，日志等公司通用业务访问的方法，降低了业务实现门槛。

开发工具

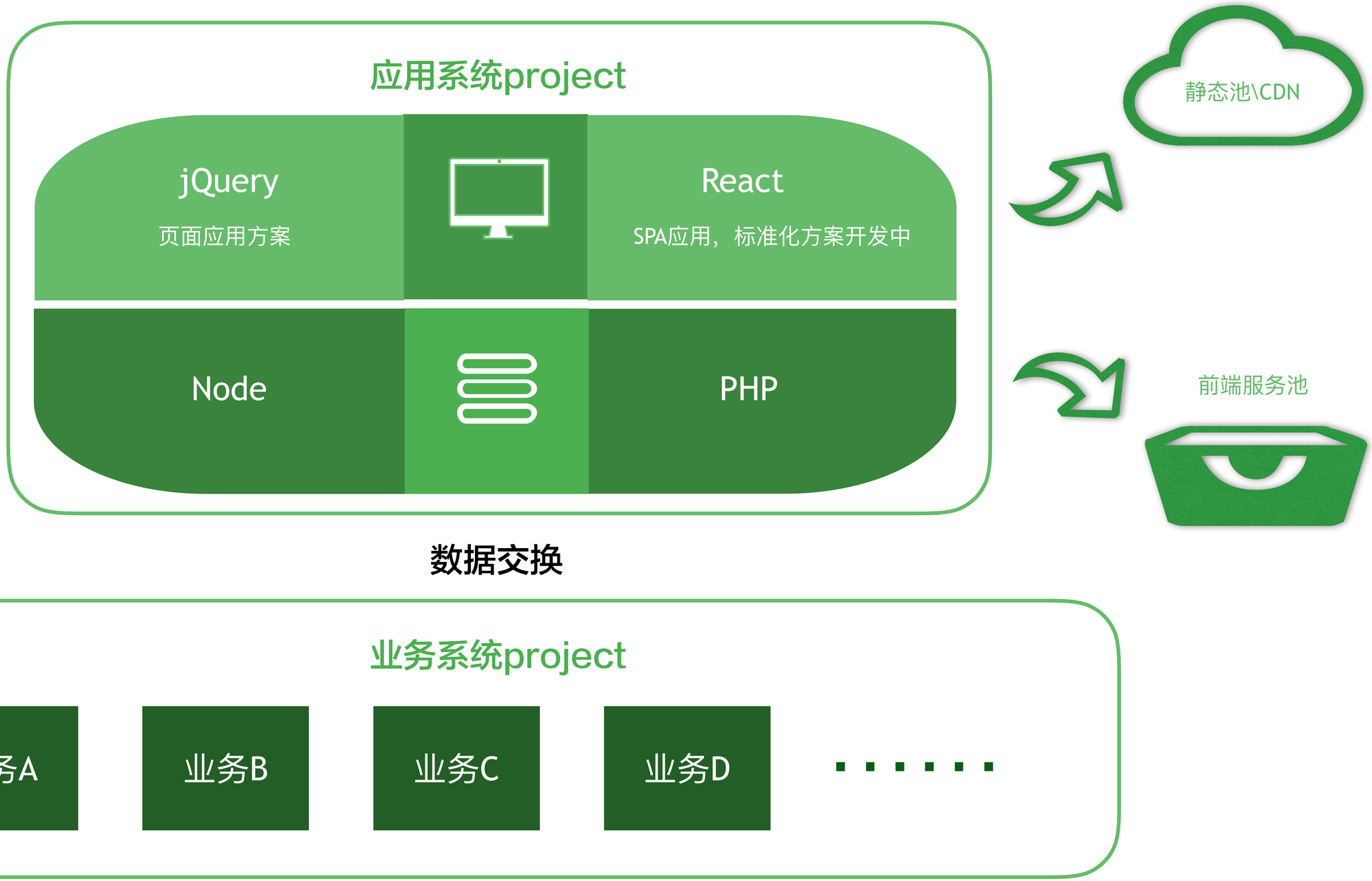
提供了开发者工具，可以方便的在web界面看到接口请求日志信息，提高联调效率。

浏览器里的方案



简单　快速　高效

体验　简化　稳定


应用系统project
jQuery
页面应用方案
React
SPA应用，标准化方案开发中
Node
PHP
静态池\CDN
前端服务池
数据交换
业务系统project
业务A
业务B
业务C
业务D
.....

总结

今天我们都聊了哪些问题

01

传统MVC

当孤立地看一个系统的时候，这个方案，快速，稳定，高效。但多方合作时易起冲突。

02

单页应用

解决了职责的问题，但开发手段单一，不解决跨系统问题，系统健康不好监控。

03

模块化发布

解决了合作和发布的问题，但对人的要求高，不易招人。不好维护。

04

应用与业务分离

通过提供框架与工具，降低开发门槛，使全团队很容易上手独立制作一个展示系统。

谢谢，Q&A