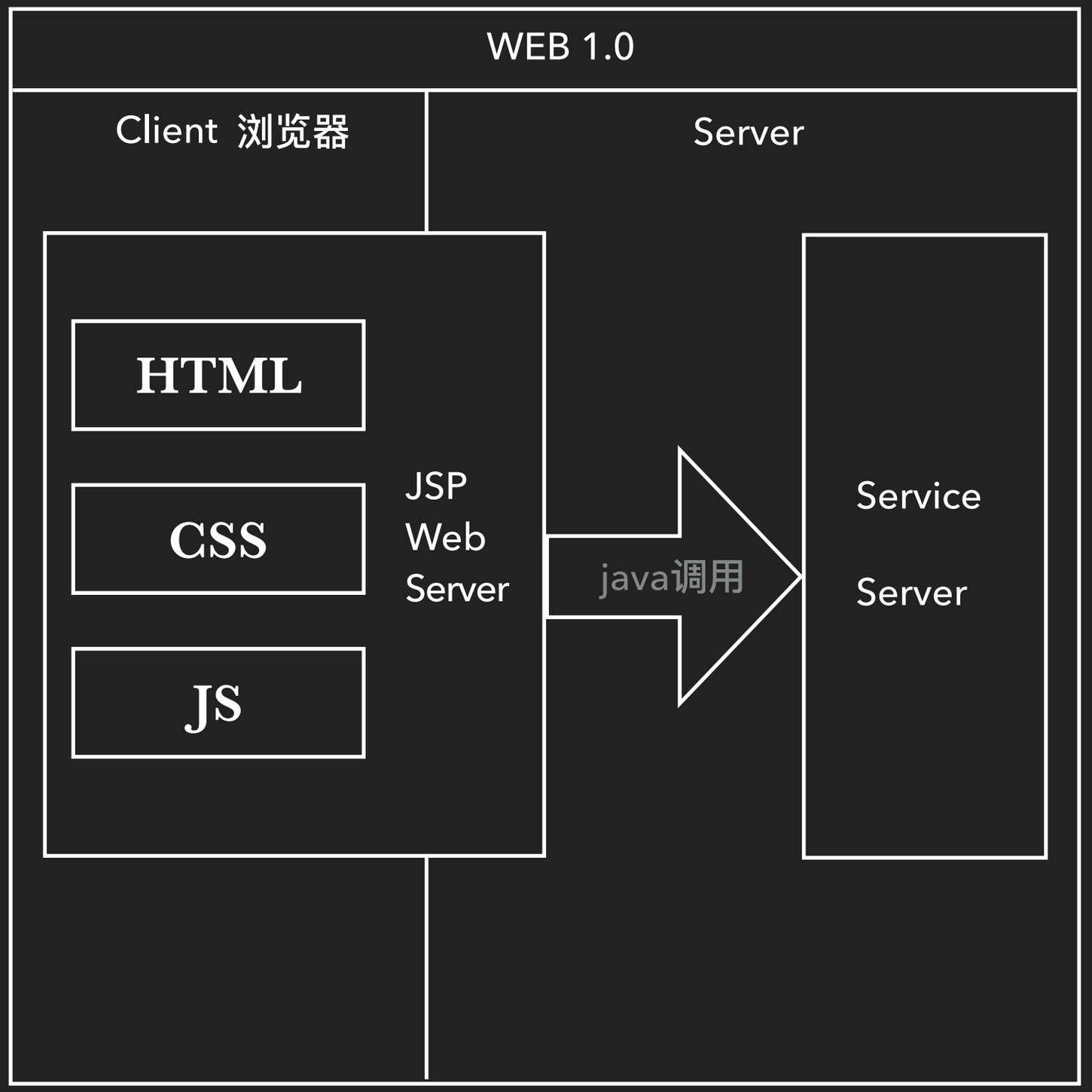
WEB研发模式演变

——友云采前后端分离实践

宋江凌

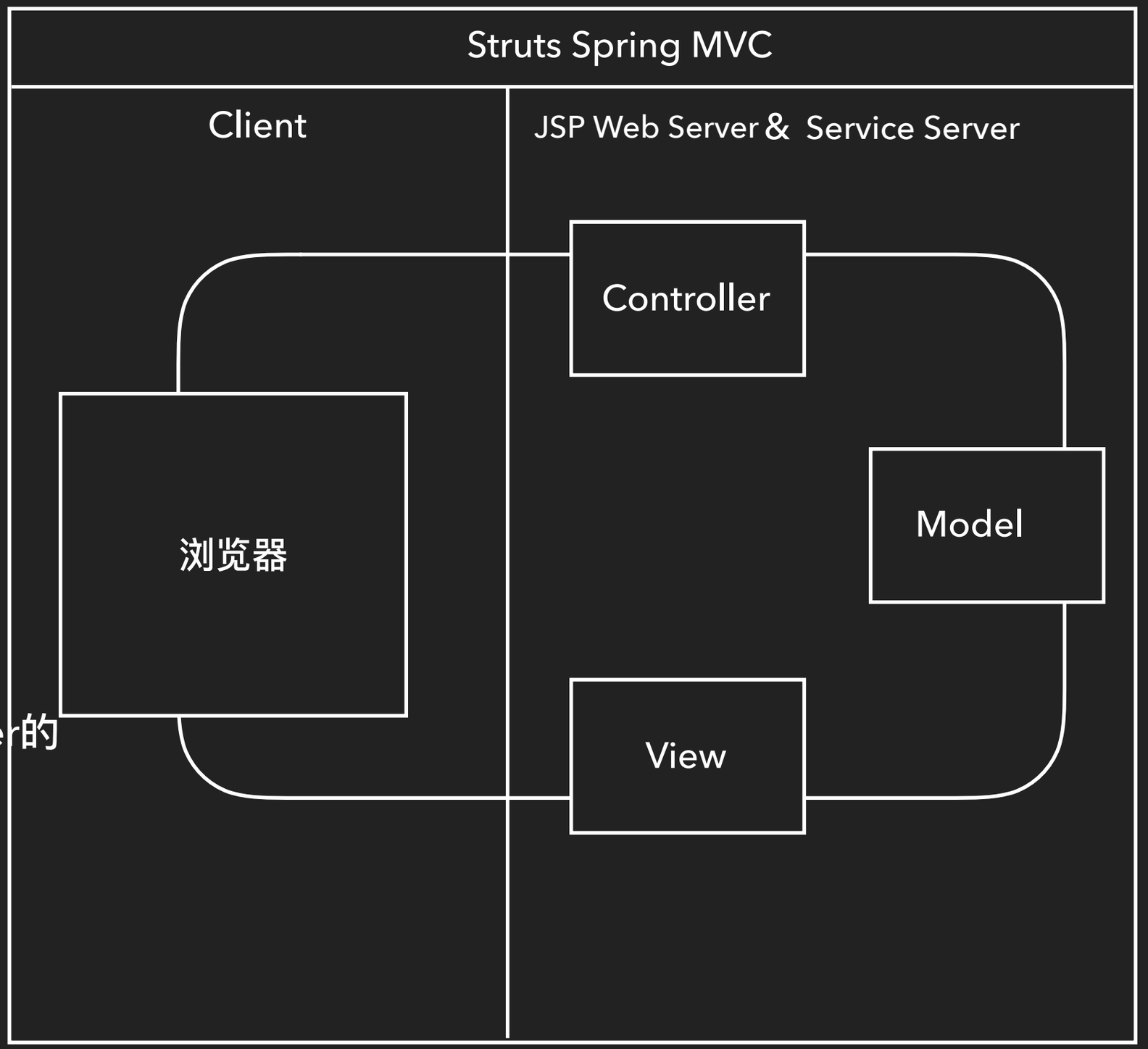
WEB 1.0

- 简单直接，后端返回什么前端展现什么
- 优点：
  - 一个tomcat就能开发
- 缺点：
  - service越来越多，依赖越来越复杂
  - JSP可维护性太差，页面大量业务代码
- 演变目的：合理分工和提升代码可维护性



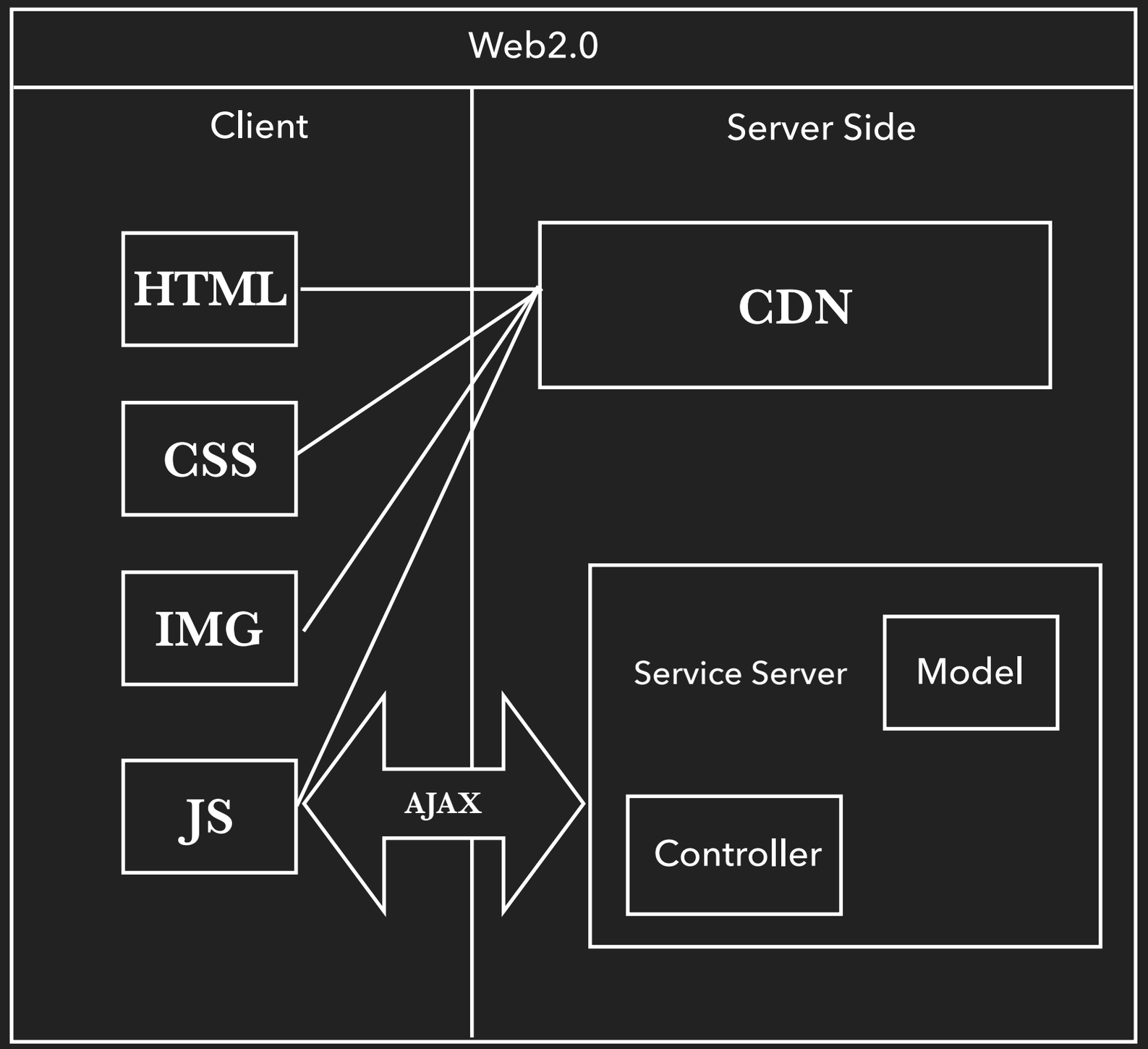
后端为主MVC

- Web Server 层架构升级，SSH
- 优点：
  - 代码分层，协作模式清晰
- 缺点：
  - 前端重度依赖后端
  - 前后端职责不清，导致Controller的灰色地带
  - 代码受限于程序员的素养



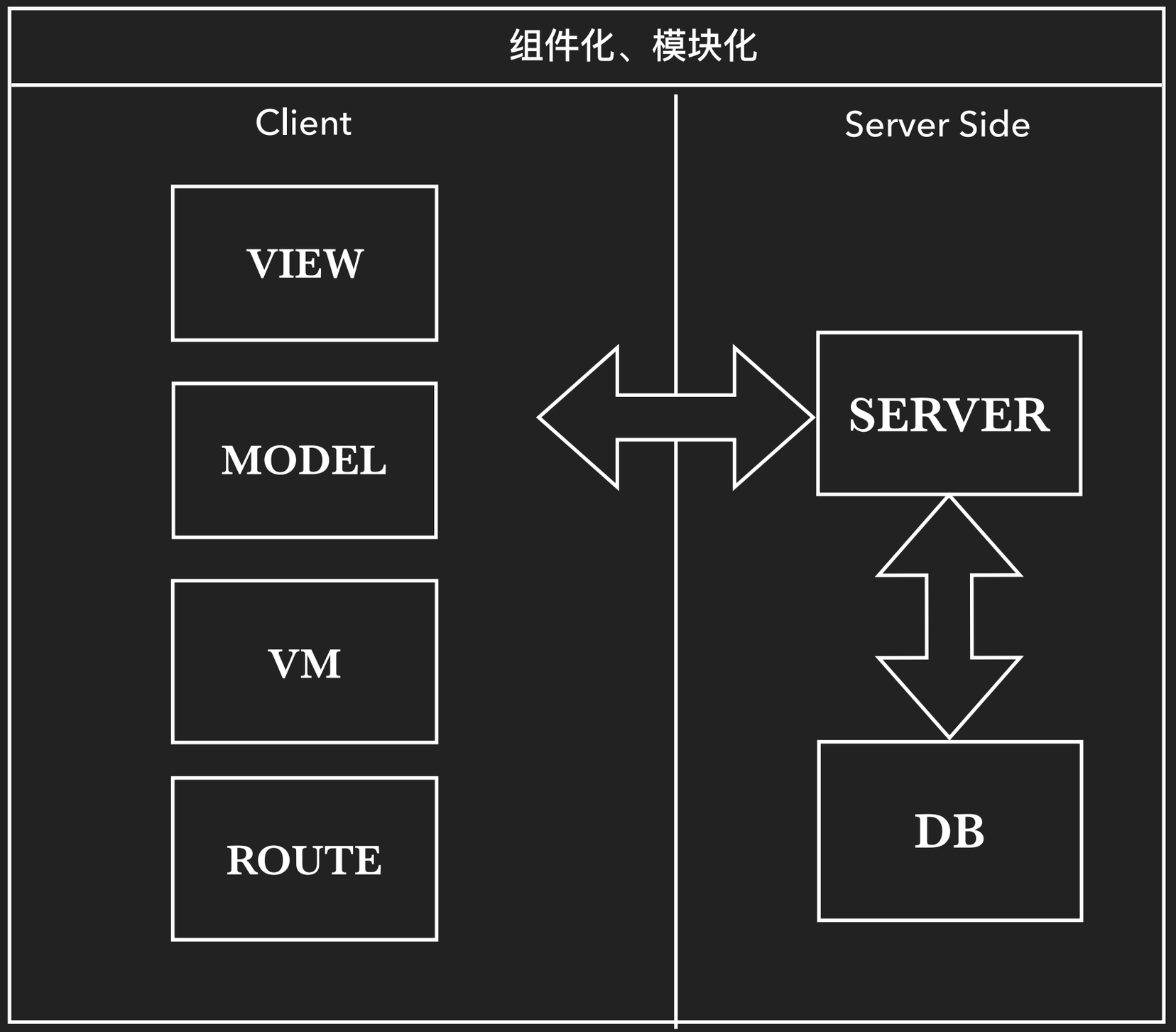
WEB2.0 AJAX

- Ajax，异步，接口
- 优点：
  - 前后端约定接口后并行开发
- 缺点：
  - 开发依赖接口的稳定程度
  - 前端大量jquery拉面式的代码
  - 代码如何组织

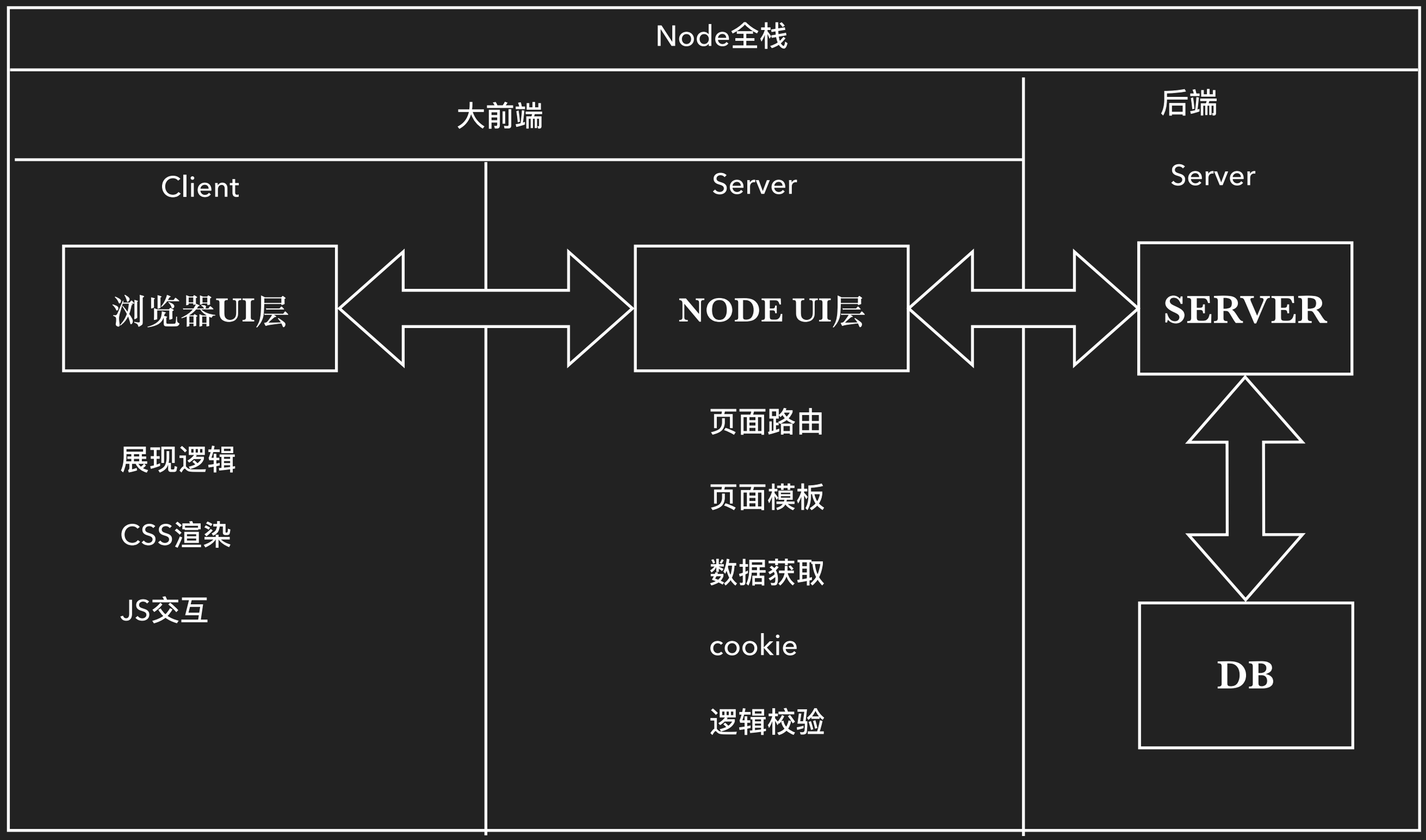


前端MV*时代

- 优点：
  - 前后端职责清晰
  - 前端代码分层合理
  - 开发部署独立
- 缺点：
  - 前后端校验不能复用
  - 异步SEO糟糕（基于百度的SEO）
  - 移动环境性能问题
  - 前端路由仍不能完全掌控，多页面应用和单页面应用的取舍



NODE全栈时代


Node全栈
大前端
后端
Client
Server
Server
浏览器UI层
NODE UI层
SERVER
展现逻辑
CSS渲染
JS交互
页面路由
页面模板
数据获取
cookie
逻辑校验
DB

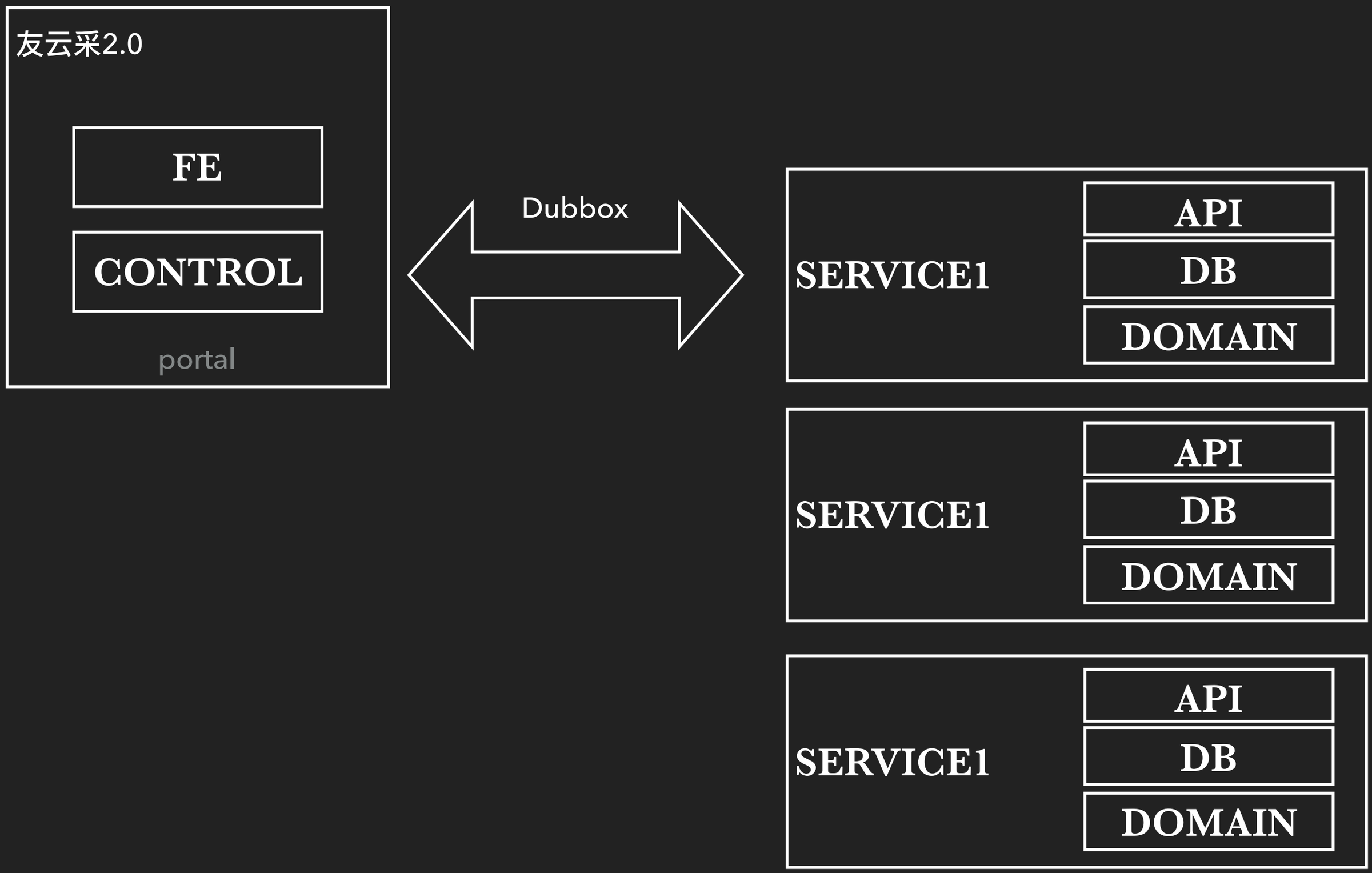
总结

- 最适合项目需求的模式才是最好的模式
- Soc（关注度分离）是一条伟大的原则，让分工更合理更高效
- 把基础学好

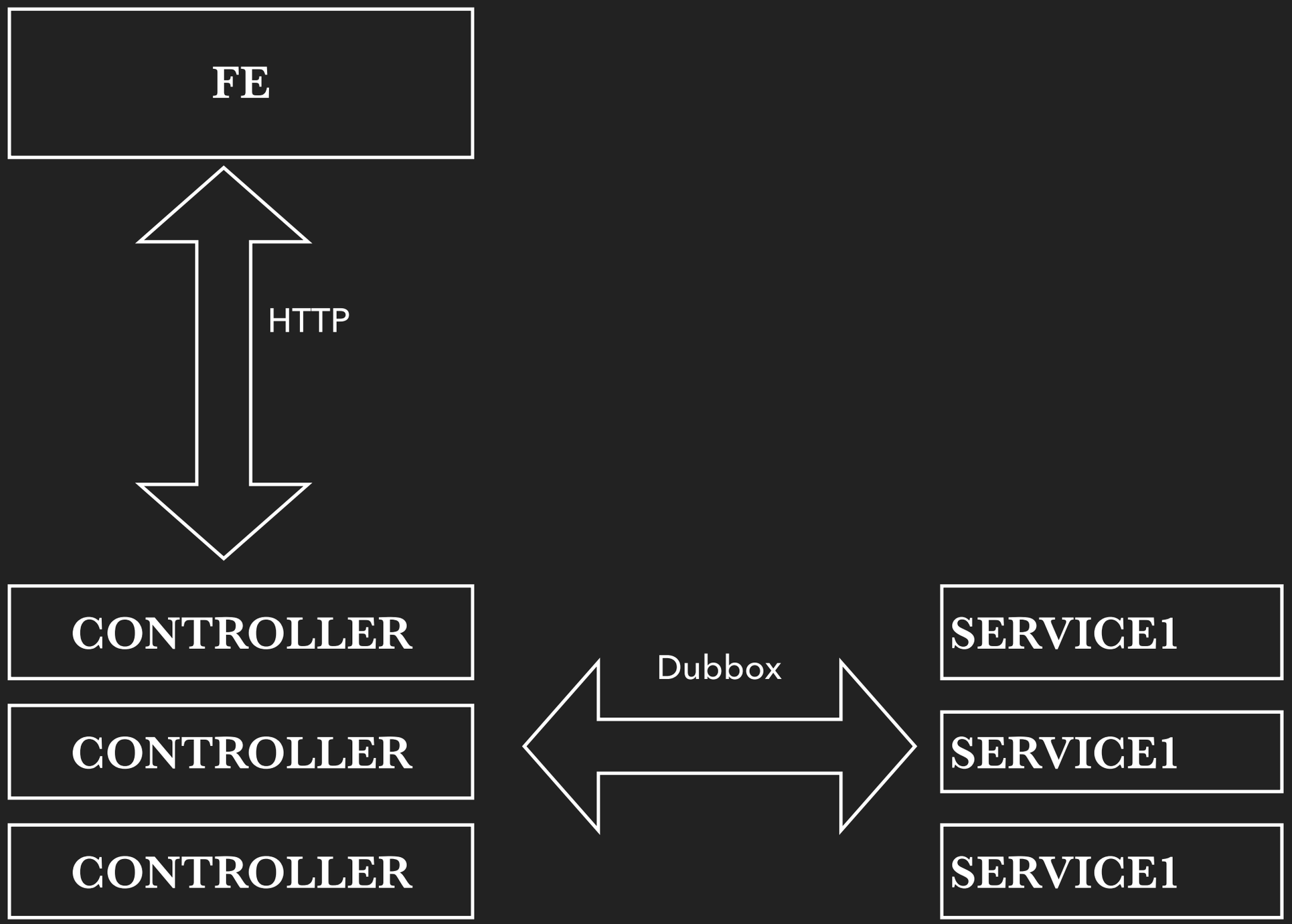
友云采前后端分离实践

- 从all in one 到 前后端分离

友云采前后端分离实践


友云采2.0
FE
CONTROL
portal
Dubbox
SERVICE1
API
DB
DOMAIN
SERVICE1
API
DB
DOMAIN
SERVICE1
API
DB
DOMAIN

招投标1.0


FE
HTTP
CONTROLLER
CONTROLLER
CONTROLLER
Dubbox
SERVICE1
SERVICE1
SERVICE1

WHY

- 分工明确，提高专注度
- 优化打包编译过程，解决js缓存的问题
- 更好的组件化
- 更好的开源社区使用
- 使用ES-Lint更好的代码规范
- 前端代码测试mocha、kamar

难点

- 如何解决跨域
- 代码如何编译和部署
- 前后端如何约定接口
- mock-server/post man
- 如何处理cookie

难点

- 如何解决跨域
  引入webpack，使用dev-proxy，解决跨域问题，同时开发时服务调用全部使用相对路径。生产环境通过nginx解决跨域问题

- 代码如何编译和部署
  使用webpack，打包到dist目录下，同时将编译好的代码提交到git库之中，jekins根据git工程将代码部署到nginx下

- 前后端如何约定接口
  引入rap.tao.org，约定接口（后续需要自己开发一套接口文档服务，把接口约定，实体描述，接口测试，mock数据，邮件变更通知都加入功能里）

- mock-server/post man
  前端使用json-server 模拟mock数据，后端使用postman开发自测，保证开发并行

- cookie问题
  在服务器端模拟一个登录页面，调用用户中心登录服务登录并回写到客户端

难点

- 开发环境如何处理cookie
  身份认证的三种方式：session-base，cookie-base，token-base。
- 如何让shiro校验通过
  - 调用用户中心登录接口
  - 回写cookie到客户端

- 谢谢

文本

- 自己的PC端UI
- 接口文档管理工具
- 定期的培训
- 成果文档化