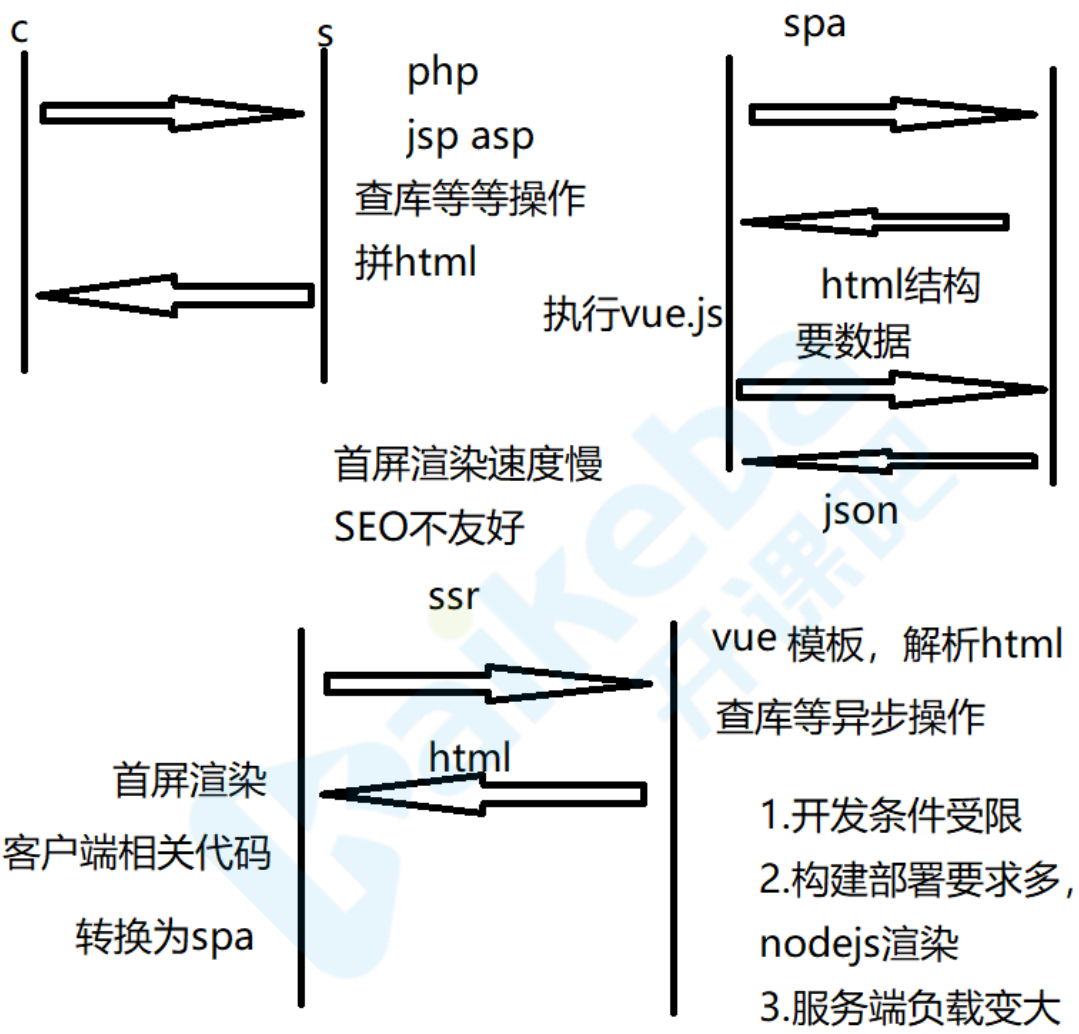
vue ssr

SSR概念

server side render

传统web渲染技术 asp.net php jsp



实现vue ssr具体过程

创建工程 vue cli 3

```
vue create ssr
```

安装依赖

渲染器vue-server-renderer

nodejs服务器express

```
npm i vue-server-render express -D
```

编写服务端启动脚本

创建一个express服务器，将vue ssr集成进来，./server/index.js

路由

安装vue-router

```
npm i vue-router -s
```

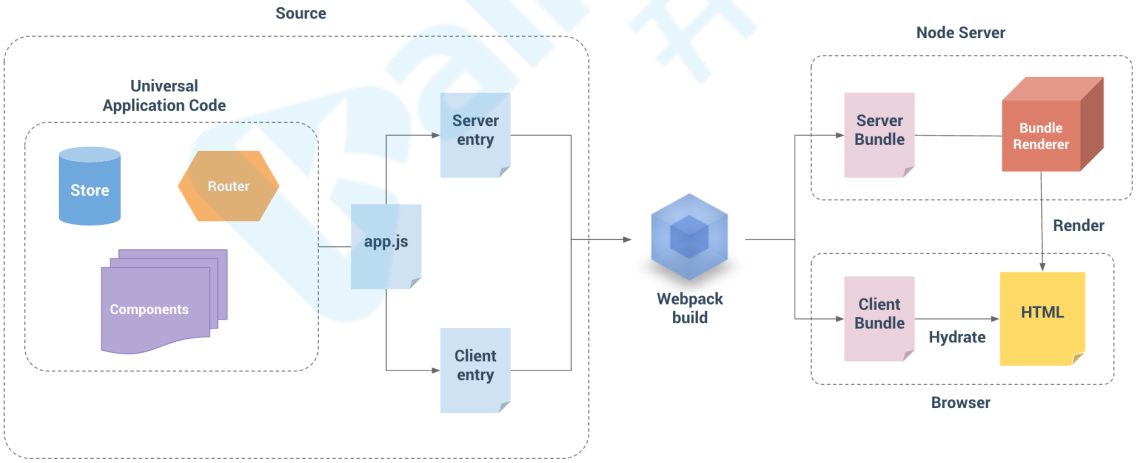
配置

创建./src/router/index.js

创建Index.vue、Detail.vue，更新App.vue

构建

构建流程



代码结构

```
src
├── components
|   ├── Foo.vue
|   ├── Bar.vue
|   └── Baz.vue
├── App.vue
├── app.js # 通用入口
├── entry-client.js # 客户端入口，仅运行于浏览器
└── entry-server.js # 服务端入口，仅运行于服务器
```

入口

app.js

服务端入口

entry-server.js

客户端入口

entry-client.js

webpack打包

安装依赖

```
npm install webpack-node-externals lodash.merge -D
```

具体配置，vue.config.js

```
const VueSSRServerPlugin = require("vue-server-renderer/server-plugin");
const VueSSRClientPlugin = require("vue-server-renderer/client-plugin");
const nodeExternals = require("webpack-node-externals");
const merge = require("lodash.merge");
const TARGET_NODE = process.env.WEBPACK_TARGET === "node";
const target = TARGET_NODE ? "server" : "client";

module.exports = {
  css: {
    extract: false
  },
  outputDir: './dist/'+target,
  configureWebpack: () => ({
    // 将 entry 指向应用程序的 server / client 文件
    entry: `./src/entry-${target}.js`,
    // 对 bundle renderer 提供 source map 支持
    devtool: 'source-map',
    // 这允许 webpack 以 Node 适用方式处理动态导入(dynamic import)，
    // 并且还会在编译 Vue 组件时告知 `vue-loader` 输送面向服务器代码(server-oriented
code)。
    target: TARGET_NODE ? "node" : "web",
    node: TARGET_NODE ? undefined : false,
    output: {
      // 此处告知 server bundle 使用 Node 风格导出模块
      libraryTarget: TARGET_NODE ? "commonjs2" : undefined
    },
    // 外置化应用程序依赖模块。可以使服务器构建速度更快，并生成较小的 bundle 文件。
    externals: TARGET_NODE
      ? nodeExternals({
          // 不要外置化 webpack 需要处理的依赖模块。
          // 可以在这里添加更多的文件类型。例如，未处理 *.vue 原始文件，
          // 你还应该将修改 `global`（例如 polyfill）的依赖模块列入白名单
```

```
          whitelist: [/\.css$/]
        })
      : undefined,
    optimization: {
      splitChunks: undefined
    },
    // 这是将服务器的整个输出构建为单个 JSON 文件的插件。
    // 服务端默认文件名为 `vue-ssr-server-bundle.json`
    plugins: [TARGET_NODE ? new VueSSRServerPlugin() : new VueSSRClientPlugin()]
  }),
  chainWebpack: config => {
    config.module
      .rule("vue")
      .use("vue-loader")
      .tap(options => {
        merge(options, {
          optimizeSSR: false
        });
      });
  }
};
```

脚本配置

安装依赖

```
npm i cross-env -D
```

定义创建脚本，package.json

```
"scripts": {
    "build:client": "vue-cli-service build",
    "build:server": "cross-env WEBPACK_TARGET=node vue-cli-service build --mode
server",
    "build": "npm run build:server && npm run build:client"
}
```

宿主文件

最后需要定义宿主文件，创建./src/index.temp.html

服务器启动文件

修改服务器启动文件，所有路由都由vue接管，使用bundle渲染器生成内容，./server/index.js