gossr

gossr 是一个用于Web开发的服务器端渲染框架(SSR)，使用 golang + V8 实现，基于Vue搭建。类似于Nuxt，Next这类SSR框架，只是它们使用Nodejs实现。

优势

- SSR框架本身的优势
  - 更好的SEO，搜索引擎爬虫可以直接抓取完全渲染的页面。
  - 更快的内容到达时间 (time-to-content)，用户将会更快速地看到完整渲染的页面，从而有更好的用户体验。
- golang + V8 实现，相比基于Nodejs的方案，有更好的性能
  - 使用golang实现服务器的框架，可以多线程调度多个V8 VM实例。
  - 在实际工程中，js往往会有内存泄漏，这个对于服务器是致命的，本框架通过设置V8 VM的生命期，生命期到后则删除该实例，从而解决了内存泄漏，保证服务器可以长时间稳定运行。
- 实现了SSR运行所需的js环境
  - 实现了CommonJS require加载规范
  - 实现了XMLHttpRequest，可以执行ajax请求
  - 实现了console.debug, console.log, console.info, console.warn, console.error
  - 没有实现SSR不推荐使用的setTimeout和setInterval方法，从而规避潜在的问题
- 支持更好的开发调试
  - SSR端和浏览器端支持统一的全局变量BASE_URL、API_BASE_URL，线上环境和开发环境代码统一
  - API_BASE_URL在开发环境自动支持api调用的转发，从而保证cookie可以正确传递
- 服务部署简单
  - golang代码编译成一个单一的可执行文件，只要在服务器部署该执行文件和webpack打包好的js脚本就可以了

编译运行

1. 安装V8环境
   - MacOS环境
     - 查看 install.v8/macos.md
   - CentOS 7环境
     - 查看 install.v8/centos7.md
   - 其他环境，暂不支持
2. golang服务器的编译
   - 进入到项目根目录
   - 运行`make`，完成编译
   - 运行`make test`，测试v8 worker
   - 运行`make clean`，清空编译的文件
3. js工程的打包
   1. 预先安装node环境
   2. `cd client`
   3. `npm install` 安装项目的依赖包

    4. `npm run build-dev`打包开发环境的包，或者`npm run build-prod`打包正式环境的包
    5. `npm run watch` 打包开发环境的包，并监控文件的变化，增量更新包内容，也就是可以热更新。
4. 运行
    1. 上述3个步骤都完成后，就可以运行服务
    2. `./gossr --config conf/goblogssr-dev.toml`命令运行，conf目录下goblogssr-prod.toml是正式环境的配置
    3. 在浏览器上访问URL: `http://localhost:8080/`，查看输出的内容

配置文件说明

- host：服务器监听的ip和端口。示例："0.0.0.0:8080"，表示监听服务器所有的ip上，端口是8080
- env： 表示服务器运行在开发环境下，还是正式环境下。"dev"表示开发环境，"prod"表示正式环境，也可以自定义环境。js定义了一个全局变量APP_ENV, 就是该值
- v8_maxcount：表示最多运行多少个V8 VM实例
- v8_lifetime：表示每个V8 VM实例的生命期，单位秒。在实例的生命期内，加载的js脚本会保持在内存，不会重新加载，在开发环境中，可以设置为0，来强制每次请求都重新创建实例。
- js_project_path：js工程的目录
- static_url_path：资源url的前缀，生成的客户端js，img，css等资源的url前缀
- is_api_delegate：服务器是否做api转发，如果为true，则会把/api前缀的前端ajax请求，转发到internal_api_host配置的服务器上的。
- internal_api_host：转发请求的host
- internal_api_ip：ssr的后端ajax请求，会强制改成该ip，这样会提高后端ajax请求的性能。此时，如果配置了internal_api_host，请求的Header头的Host会设置成该值。
- internal_api_port：上述配置的api请求服务器的port
- template_name：输出html页面的模版，模版目录是 client/server_dist/template
- client_cookie：如果不为空，gossr服务器会生成一个以它命名的生命期很长的cookie，可以做为客户端的标识id
- redirect_onerror：如果不为空，生成页面的js脚本如果发生错误，则请求返回302 页面跳转的响应
- ssr_ctx：ssr框架会根据配置的header，获取请求对应的header值，并传到js脚本里，脚本里可以通过context.ssrCtx访问对应的header值。并且，后端ajax请求，也会自动带上这些header值。缺省会包含Cookie这个header
- template_vars：模版的变量定义，会把脚本生成的state.meta对象对应的值，映射到模版里输出。type为js表示变量的内容是js脚本；type为html表示变量的内容是html内容，不会做字符<、>等的逃逸。其他的type会做内容的逃逸，并且不必在配置项中列出。

部署文件列表

```
├── gossr
├── conf/
└── client/
    ├── dist/
    ├── dist_server/
    └── node_modules/
        └── vue-server-renderer/
```