Node日志

一、日志级别

只使用 `FATAL`、`ERROR`、`WARN`、`INFO` 和 `DEBUG` 等级。

> 1、FATAL - 导致程序退出的严重系统级错误，不可恢复，当错误发生时，系统管理员需要立即介入，一般应用代码 **不** 使用。
>
> 2、ERROR - 运行时异常以及预期之外的错误，也需要立即处理，但紧急程度低于FATAL,当错误发生时，影响了程序的正确执行。需要注意的是这两种级别属于服务自己的错误，需要管理员介入，**用户输入出错不属于此分类**，请求后端、读文件、数据库等超时、返回错误结构，属于ERROR；
>
> 3、WARN - 预期之外的运行时状况，表示系统可能出现问题。对于那些目前还不是错误，然而不及时处理也会变成错误的情况，也可以记为WARN，如磁盘过低。
>
> 4、INFO - 有意义的事件信息，记录程序正常的运行状态，比如收到请求，成功执行。通过查看INFO,可以快速定位WARN，ERROR, FATAL。INFO不宜过多，通常情况下不超过 DEBUG 的10%。
>
> 5、DEBUG - 与程序运行时的流程相关的详细信息以及当前变量状态。

二、脚手架日志组件

软件二部前端脚手架 sword 使用的默认为 log4js 进行日志记录。框架提供完善的日志系统，使用log4js实现，日志在线上排查问题时起至关重要的作用，所以我们需要合理的添加日志，不能滥用和乱用日志，否则记录日志就无意义。

1.日志分类

除了日志级别外，我们排查问题时还会查看对应的日志分类，分类可以让我们能够缩小分析范围，框架现目前内置了几种日志

- **service**：和后端服务交互的日志，包含请求和响应日志，每个日志都带有`request-id`，建议项目组后端使用这个`request-id`，便于排查问题。
- **cluster-worker**：`cluster`工作进程相关日志，主要记录了进程的未捕获错误。
- **cluster-master**：`cluster`主进程相关日志，主要记录了工作进程重启情况、系统重启频繁挂掉的日志。
- **access**：记录浏览器访问`nodejs`每次请求的日志，包含url、method、user-agent、响应时间等。
- **app**：记录app的一些运行日志，如启动日志和业务运行日志

> 以上除了app类别的日志框架内部都自动做了处理，不需要再项目中记录，它们都属于系统日志。

2.app日志

在项目中我们可以使用`sword.logger.xx`来记录日志，对一些关键业务节点记录日志埋点，这些日志记录的类别都是`app`这个类别。

```
sword.logger.info('日志内容');
```

如果我们想要记录一些特殊类别的日志，我们可以使用`sword.getLogger(category)`创建一个logger，然后使用。

```
const customLogger = sword.getLogger('custom');
customLogger.info('日志内容');
```

3. logger对象

logger对象包含和日志级别对应的方法

```
logger.trace();
logger.debug();
logger.info();
logger.warn();
logger.error();
logger.fatal();
```

4.日志输出

在开发环境下，日志会输出到控制台，便于查看

在生产环境下，日志默认会输出到根目录下的logs`目录中，文件名为app.log.YYYY-MM-DD，每天都会生成一个文件。

5. 问题排查

对于一般能够复现的问题，我们可以通过浏览器接口响应来确定是哪里除了问题，但是如果是无法复现的问题我们就需要合理的利用日志来排查了。

比如客户反馈某个页面在一段时间出现请求失败的提示，但是过后又正常了，这种情况我们就可以通过日志排查。

首先知道是哪个页面，就知道具体是哪些api的问题，接着排查这些api使用到的服务有没有报错(可以通过服务的api name定位)，如果报错，定位到问题，如果没有错误日志，再查一下app类别的日志有没有错误日志，如果有排查一下这些错误日志是不是该页面的问题，就能快速的定位了。

三、更多内容

可以参考现有 sword 项目 git 文档。