Eve-plugin-hoirst2 使用-浏览器

作者：Yang 发布时间：2021-12-30 阅读数：1,437

微信扫一扫
分享给朋友 / 朋友圈

收录于话题

#eve-plugin-hoirst2 1 #Hoirst 1

> 特别说明：本 Eve-plugin-hoirst2 系列文章 v1.1版 已经有了不同于之前版本的变化，v1.1 版本基于 Eve 的 v0.9.0 版本，v1.1 版本 Hoirst 的使用方式和之前版本的使用方式不同，v1.0 版本的实现，请参阅历史版本最早期的文章。
>
> 更新说明：https://github.com/lt-hoirst/eve-plugin-hoirst-start

▲ 可视化管理

1.1 编写 provider 的 接口

这里我们编写了 provider接口文件，用于通过浏览器访问 Eve 数据 ，也可以通过命令行执行。

1.1.1 创建 provider

编写创建 provider文件，创建 provider：

```
import { Injectable } from "eve-plugin-hoirst";
import { Inject, provide, Provider } from "eve-core";
import { Logger, ILogger } from "eve-logger";
var _dbPrefixKey = "hoirst:";
var _dbPrefixFunc = function (name) {
    return _dbPrefixKey + (name.indexOf("hoirst:") == 0 ? name.substr(7) : name);
};
var _dbPrefixDb = function (db, name, cb) {
    var _name = _dbPrefixFunc(name);
    if (!db.hasOwnProperty(_name)) {
        db.create(_name, cb);
    } else {
        cb();
    }
};
var _getDbFunc = function (db, name) {
    var _name = _dbPrefixFunc(name);
    return {
        find: function(query, cb) {
            db.find(_name, query, cb);
        },
        get: function(id, cb) {
            db.get(_name, id, cb);
        },
        insert: function(data, cb) {
            db.insert(_name, data, cb);
        },
        update: function(id, data, cb) {
            db.update(_name, id, data, cb);
        },
        remove: function(id, cb) {
            db.remove(_name, id, cb);
        }
    }
};
var _getDbFuncs = function (db, names) {
    var ret = {};
    for (var i = 0; i < names.length; i++) {
        var name = names[i];
        var _name = _dbPrefixFunc(name);
        ret[_name] = _getDbFunc(db, name);
    }
    return ret;
};
@Injectable()
@provide(Provider)
export class _dbProvider {
    @Inject(Logger) _log: ILogger;
    _db: any;
    _dbFuncs: {[name: string]: Function};
    _dbPrefixKey: string;
    _dbPrefixFunc: Function;
    _dbPrefixDb: Function;
    _getDbFunc: Function;
    _getDbFuncs: Function;
    constructor(db: any) {
        this._db = db;
        this._dbPrefixKey = _dbPrefixKey;
        this._dbPrefixFunc = _dbPrefixFunc;
        this._dbPrefixDb = _dbPrefixDb;
        this._getDbFunc = _getDbFunc;
        this._getDbFuncs = _getDbFuncs;
        this._dbFuncs = this._getDbFuncs(this._db, ["hoirst:hoirst", "hoirst:hoirstcfg"]);
        this._log.info("hoirst provider initialized");
    }
    getHoirst() {
        return this._dbFuncs["hoirst:hoirst"];
    }
    getHoirstcfg() {
        return this._dbFuncs["hoirst:hoirstcfg"];
    }
}
export default _dbProvider;
```

现在这个 provider 已经可以使用了。

1.1.2 创建 load

Hoirst 的功能，只要注册 load 即可实现。

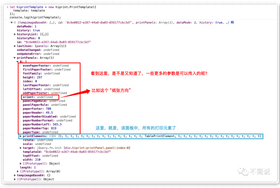
```
import { Inject } from "eve-core";
import { Logger, ILogger } from "eve-logger";
import { Injectable, Provider } from "eve-plugin-hoirst";
import { _dbProvider } from "./_dbprovider";
// 注册 load 函数
export function load(_log: ILogger, _dbProvider: _dbProvider) {
    var _getHoirst = _dbProvider.getHoirst();
    var _getHoirstcfg = _dbProvider.getHoirstcfg();
    var _log = _log.getLogger("hoirst");
    var _db = _dbProvider._db;
    var _dbFuncs = _dbProvider._dbFuncs;
    var _dbPrefixKey = _dbProvider._dbPrefixKey;
    var _dbPrefixFunc = _dbProvider._dbPrefixFunc;
    var _dbPrefixDb = _dbProvider._dbPrefixDb;
    var _getDbFunc = _dbProvider._getDbFunc;
    var _getDbFuncs = _dbProvider._getDbFuncs;
    var _hoirst = _getHoirst();
    var _hoirstcfg = _getHoirstcfg();
    var _log = _log.getLogger("hoirst");
    var _db = _db;
    var _dbFuncs = _dbFuncs;
    var _dbPrefixKey = _dbPrefixKey;
    var _dbPrefixFunc = _dbPrefixFunc;
    var _dbPrefixDb = _dbPrefixDb;
    var _getDbFunc = _getDbFunc;
    var _getDbFuncs = _
```

▲ 带参数的模板对象

嗯嗯，很明显，这几个参数是有作用的呢。

我们再试试，传入一个 模板json：

```
let template = { panels:[] }; // 设置代回调模板，初始数组还是一空的~
let hiprintTemplate = new hiprint.PrintTemplate({
  template: template
});
console.log(hiprintTemplate);
```



▲ 带模板json的模板对象

好了，相信看到这里，对于 模板对象 的 结构 你一定已经有所了解了。接下来就是 API 了。

4.2 模板对象的API

看了几张图，我们还没有看到常用 API，如 `setPaper`、`zoom`、`getJson` 等等，那它应该在哪里呢？别它呢，没找到就先东点西点看看看 😅，当我们点开最后一个 `[[Prototype]]` 时，你将发现 新大陆：

▲ 模板对象-新大陆-

如上图，我们常用的API好像都在这里了呢。

新大陆已发现，不探索一下，怎么行呢？

既好比在 新大陆 看到了很多 树，那 树 上有些什么 怎么看 呢？

4.2.1 查看API源码

我们以 `setPaper` API为例，带大家来 一探究竟：

▲ setPaper示例

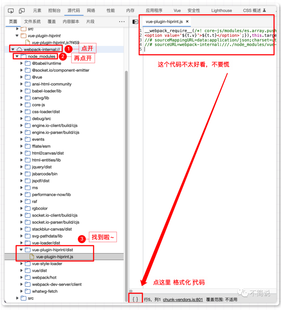
我们可以清楚看到这个 `setPaper` 需要传两个参数，然后后面又一个 `[[FunctionLocation]]`，它后面个 链接 的形式，是可以点击，`FunctionLocation` 翻译过来就是 函数位置，那么很明显了呢，我们就要从这里开始~

▲ 无法查看(出师不利？)

不要慌，想想我们 主要的目的 是 看什么？ 清楚了目的是 `vue-plugin-hiprint`，我们找一找呗~

▲ 凡事不要慌

皇天不负有心人，可算在这里找到你了：



▲ 发现目标

当我们 格式化代码 后，哈哈，好看多了嘛。我们再回到先前的 `[[FunctionLocation]]`，再次点击这个 链接：

▲ 再次点击链接

欧克欧克，终于看到了 `API` 源码 啦~

▲ 终于看到了

好了，关于 如何查看有什么API 就到这里了，至于源码做了些什么，就 需要各位自行探索 了。

此处可以说是 网页调试 相关的一些内容，如果你不会 浏览器调试技巧，那么赶紧科普科普去吧~

总结

- 本篇内次 整述 了 provider 和 设计器 的构建，希望大家能够很清楚的去 构建渲染
- 了解 vue3 的 组合式函数 使用场景
- 通过常用的 API 引导大家去探索 模板对象 的 结构
- 手把手 的教大家 如何去探索 更多的 模板对象 API

进阶篇 到此结束，有些不清楚的知识点，大家 一定要学会自己查阅相关资料。

欢迎各位码友 转发 及 留言反馈，觉得不错就 点个赞 再走哈~

收录于合集 #vue-plugin-hiprint 4

上一篇
【vue-plugin-hiprint】使用-入门篇

下一篇
【vue-plugin-hiprint】使用-打印篇

喜欢此内容的人还喜欢

别总写代码，这130个网站比涨工资都重要！
java架构师

用链表实现栈，并且用栈实现简单的加减乘除运算
亿笑横城

深入理解错误
写不出来！

主题模板
python