

```

1
2 import { WorkerClass } from './workerProxy_2.js';
3 import { Job } from './Job_1.js';
4
5 ///////////////////////////////////////////////////////////////////
6 //
7 // 整合所有的 worker
8 //
9 ///////////////////////////////////////////////////////////////////
10
11 class WorkerManager {
12
13     static get_instance(workerCommand) {
14         if (WorkerManager.instance == null) {
15             WorkerManager.instance = new WorkerManager(workerCommand);
16         }
17         return WorkerManager.instance;
18     }
19     //-----
20     constructor(root) {
21
22         // 與外界橋接的橋樑
23         this.root = root;
24
25         //
26         this._;
27
28         this.settings;
29
30         this.environment;
31
32         // 記錄有多少在線的 worker
33         this.workers = new Set();
34
35         // 正在閒置中的 workers(等死中)
36         // this.idleWorks = new Set();
37         this.jobList = [];
38
39         // 因應各種環境引入不同的 worker
40         // 一個重要的設計點
41         this.workerClass;
42
43         //-----
44         this._getSettings();
45     }
46     //-----
47     _getSettings() {
48         debugger;
49
50         this.settings = this.root.settings;
51
52         this._ = this.root._;
53
54         if (this._ == null) {
55             throw new Error('no import _');
56             // console.log('no connection with _');
57         }
58
59         if (this._.$extension1 == null) {
60             throw new Error("need import _extension1");
61         }
62

```

```

63      // 取得環境
64      this.environment = this._.$extension1.info.environment;
65
66      //-----
67      // 取得適合當前環境下的 workerClass
68
69      switch (this.environment) {
70          case "nodejs":
71              this.workerClass = WorkerClass.NodeJsWorkerProxy;
72              break;
73          default:
74              this.workerClass = WorkerClass.WebWorkerProxy;
75              break;
76      }
77  }
78  //-----
79  addJob(args) {
80      // debugger;
81      // console.log('-----');
82      console.log('WorkerManager > 加入新工作');
83
84      let job = new Job(this, args);
85
86      // 把任務加入
87      this.jobList.unshift(job);
88      //-----
89      // 檢查是否有 idle worker
90      this.noticeWorkers_checkJob();
91
92      return job.promise();
93  };
94  //-----
95  // 預先不會在一開始啟動 workers
96  // 通常只有在有指令後才會有 workers 待命
97  // 不過可以事先就請 workers 待命
98  initWorkers(count) {
99      debugger;
100
101      const min_workers = this.settings.min_workers;
102      const max_workers = this.settings.max_workers;
103
104      if(count > max_workers){
105          throw new Error('initWorkers.count > max_workers(${max_workers})');
106      }
107
108      for (let i = 0; i < count; i++) {
109          debugger;
110
111          if(this.workers.size >= max_workers){
112              break;
113          }
114
115          let employment = false;
116          if (this.workers.size < min_workers) {
117              // 正職還有缺額
118              employment = true;
119          }
120          new this.workerClass(this, employment);
121      }
122  }
123  //-----
124  // 針對 node.js

```

```

125   terminateAllWorkers(){
126
127   }
128   //-----
129   // 請工作人員注意是否有工作進來
130   // worker: {worker: 由 worker 呼叫, null: 由 manager 呼叫}
131   noticeWorkers_checkJob(worker) {
132       if (worker == null) {
133           // console.log('check by manager');
134       } else {
135           console.log('check by worker(%s)', worker.id);
136       }
137
138
139       while (this.jobList.length > 0) {
140
141           console.log('still have jobs(%s)', this.jobList.length);
142
143           // 盡可能招募 worker 來接工作
144           let w = this._checkIdleWorker();
145
146           if (w) {
147               let job = this.jobList.pop();
148               w.takeJob_callByManager(job);
149           } else {
150               // 若找不到可用的 worker 作罷
151               break;
152           }
153       }
154
155   }
156
157   //-----
158   // 新增 worker(由 worker 自己通報)
159   addWorker(workerProxy) {
160       this.workers.add(workerProxy);
161   }
162   //-----
163   // 移除指定的 worker(由 worker 自己通報)
164   removeWorker(workerProxy) {
165       this.workers.delete(workerProxy);
166   }
167   //-----
168   // worker 想取得 job
169   getJob_callByWorker = function () {
170       let job = null;
171
172       console.log('有(%s)項工作待領', this.jobList.length);
173
174       if (this.jobList.length > 0) {
175           job = this.jobList.pop();
176       }
177
178       return job;
179   }
180   //-----
181   // 檢查是否有空閒的 worker
182   _checkIdleWorker() {
183       // debugger;
184
185       console.log('manager find worker');
186

```

```

187     const max_workers = this.settings.max_workers;
188     const min_workers = this.settings.min_workers;
189
190     let idleWork;
191     // 找尋空閒中的 worker
192     let idleWorks = this.findIdleWorkers();
193
194     if (idleWorks.length > 0) {
195
196         // 優先找正職者
197         idleWorks.some(function (w) {
198             if (w.employment) {
199                 idleWork = w;
200                 return true;
201             }
202         });
203
204         idleWork = idleWork || idleWorks[0];
205
206         // console.log('manager find idle worker(${idleWork.id})');
207
208         return idleWork;
209     }
210     //-----
211
212     // 沒有空閒中的 worker
213     if (this.workers.size < max_workers) {
214         // 沒有閒置的 worker
215         // 但已有的 worker 數量尚未達上限
216
217         let employment = false;
218
219         if (this.workers.size < min_workers) {
220             // 正職還有缺額
221             employment = true;
222         }
223
224         // console.log('manager employment new worker(employment: ${employment})');
225
226         new this.workerClass(this, employment);
227     } else {
228         console.log('manager no find worker');
229     }
230 }
231 //-----
232 // 找出閒置中的 worker
233 findIdleWorkers() {
234     let workers = Array.from(this.workers);
235
236     workers = workers.slice();
237
238     workers = workers.filter(function (w) {
239         if (w.isReady2TakeJob()) {
240             return true;
241         }
242     });
243     return workers;
244 }
245 //-----
246 // 取得需要的資訊
247 getAllworkersInfo() {

```

```
249     let all = [];
250     let idle = [];
251     this.workers.forEach(function (w) {
252         all.push(w.id);
253
254         if (!w.isBusy()) {
255             idle.push(w.id);
256         }
257     });
258
259     let jobCount = this.jobList.length;
260     //-----
261     return {
262         all: all,
263         idle: idle,
264         jobCount: jobCount,
265     }
266 }
267 //-----
268 }
269
270 WorkerManager.instance;
271
272 //=====
273 WorkerClass.GModules["WorkerManager"] = WorkerManager;
274 Job.GModules["WorkerManager"] = WorkerManager;
275
276 WorkerManager.GModules = {};
277
278 export { WorkerManager };
```