

Example07~13

CCR Coordination Primitives CCR

Example 07: Create a Port<int> and then Activate a Single Item Receiver

```
1
2
3
4
5 using System;
6 using System.Collections.Generic;
7 using System.Text;
8 using Microsoft.Ccr.Core;
9
10 namespace Example
11 {
12     class Program
13     {
14         // Example 07: Create a Port<int> and then Activate a Single Item Receiver
15
16         // create dispatcher and dispatcher queue for scheduling tasks
17         static Dispatcher dispatcher = new Dispatcher();
18         static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);
19
20         static void Main(string[] args)
21         {
22             /// <summary>
23             /// 示範主動抓取Port內的資料方式
24             /// </summary>
25
26             var port = new Port<int>();
27             Arbiter.Activate(_taskQueue, // Arbiter.Activate: 使用某個 DispatcherQueue 來執行一系列的
28 Tasks.
29             Arbiter.Receive(
30                 true, // false: execute one item then un-register from the port
31                 port,
32                 item => Console.WriteLine(item)
33             )
34             );
35
36             // post item, so delegate executes
37             port.Post(5);
38             port.Post(7); // 平行處理
39         }
40     }
41 }
```

42 Example 08: The Same with Example 07, Passing 43 Arbiter Class Factory Receiver

```
44 using System;
45 using System.Collections.Generic;
46 using System.Text;
47 using Microsoft.Ccr.Core;
48
49 namespace Example
50 {
51     class Program
52     {
53         // Example 08: The Same with Example 07, Passing Arbiter Class Factory Receiver
54         static void Main(string[] args)
55         {
56             /// <summary>
57             /// 間接啟動抓取資料
58             /// </summary>
59
60             var port = new Port<int>();
61
62             // alternate version that explicitly constructs a Receiver by passing
63             // Arbiter class factory methods
64             var persistedReceiver = new Receiver<int>(
65                 true, // persisted, false: execute one time
66                 port,
67                 null, // 新增項目: no predicate
68                 new Task<int>(item => Console.WriteLine(item)) // task to execute
69             );
70
71             Arbiter.Activate(_taskQueue, persistedReceiver);
72
73             // post item, so delegate executes
74             port.Post(5);
75             port.Post(7);
76         }
77
78         // create dispatcher and dispatcher queue for scheduling tasks
79         static Dispatcher dispatcher = new Dispatcher();
80         static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);
81     }
82 }
```

83 Example 09: Choice Arbiter

```
84 using System;
85 using System.Collections.Generic;
```

```

86 using System.Text;
87 using Microsoft.Ccr.Core;
88
89 namespace Example
90 {
91     class Program
92     {
93         // Example 09: Choice Arbiter
94         static void Main(string[] args)
95         {
96             // create a simple service listening on a port
97             ServicePort servicePort = SimpleService.Create(_taskQueue); // 呼叫「ServicePort Create」
98
99             // Update the state of SimpleService
100            UpdateState _state = new UpdateState(); // public string State;
101            _state.State = "Test";
102            //servicePort.Post(_state); // success or failure
103
104            // create request
105            GetState get = new GetState();
106
107            // post request
108            servicePort.Post(get);
109
110            // use the extension method on the PortSet that creates a choice
111            // given two types found on one PortSet. This a common use of
112            // Choice to deal with responses that have success or failure
113            Arbiter.Activate(_taskQueue,
114                get.ResponsePort.Choice(
115                    s => Console.WriteLine(s), // delegate for success
116                    ex => Console.WriteLine(ex) // delegate for failure
117                ));
118        }
119
120        // create dispatcher and dispatcher queue for scheduling tasks
121        static Dispatcher dispatcher = new Dispatcher();
122        static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);
123
124        #region Example 09: function
125        /// <summary>
126        /// Example 09
127        /// </summary>
128        public class ServiceOperation
129        {
130            public PortSet<string, Exception> ResponsePort = new PortSet<string, Exception>(); // 建立ResponsePort的PortSet
131        }
132
133        public class Stop : ServiceOperation
134        {
135        }
136        }
137

```

```

138     public class UpdateState : ServiceOperation
139     {
140         public string State;
141     }
142
143     public class GetState : ServiceOperation
144     {
145     }
146
147     /// <summary>
148     /// PortSet that defines which messages the services listens to
149     /// </summary>
150     public class ServicePort : PortSet<Stop, UpdateState, GetState>
151     {
152     }
153     /// <summary>
154     /// Simple example of a CCR component that uses a PortSet to abstract
155     /// its API for message passing
156     /// </summary>
157     public class SimpleService
158     {
159         ServicePort _mainPort;
160         DispatcherQueue _taskQueue;
161         string _state;
162
163         public static ServicePort Create(DispatcherQueue taskQueue)
164         {
165             var service = new SimpleService(taskQueue);
166             service.Initialize();
167             return service._mainPort;
168         }
169
170         private void Initialize()
171         {
172             // using the supplied taskQueue for scheduling, activate two
173             // persisted receivers, that will run concurrently to each other,
174             // one for each item type
175             Arbiter.Activate(_taskQueue,
176                 Arbiter.Receive<UpdateState>(true, _mainPort, UpdateHandler),
177                 Arbiter.Receive<GetState>(true, _mainPort, GetStateHandler)
178             );
179         }
180
181         private SimpleService(DispatcherQueue taskQueue)
182         {
183             // create PortSet instance used by external callers to post items
184             _mainPort = new ServicePort();
185
186             // cache dispatcher queue used to schedule tasks
187             _taskQueue = taskQueue;
188         }
189

```

```

190     void GetStateHandler(GetState get)
191     {
192         if (_state == null)
193         {
194             // To demonstrate a failure response,
195             // when state is null will post an exception
196             get.ResponsePort.Post(new InvalidOperationException()); //
197             InvalidOperationException 內建錯誤回傳函式
198             return;
199         }
200
201         // return the state as a message on the response port
202         get.ResponsePort.Post(_state);
203     }
204     void UpdateHandler(UpdateState update)
205     {
206         // update state from field in the message
207         _state = update.State;
208
209         // as success result, post the state itself
210         update.ResponsePort.Post(_state);
211     }
212 }
213 #endregion
214 }
215 }

```

216 Example 10: Joint

```

217 using System;
218 using System.Collections.Generic;
219 using System.Text;
220 using Microsoft.Ccr.Core;
221
222 namespace Example
223 {
224     class Program
225     {
226         // Example 10: Joint
227         static void Main(string[] args)
228         {
229             /// <summary>
230             /// Using Joint to listen the same port will be a race, It cant guarentee that the Arbiter
231             will run.
232             /// </summary>
233
234             var portDouble = new Port<double>();
235             var portString = new Port<string>();
236

```

```

237     // activate a joined receiver that will execute only when one
238     // item is available in each port.
239     Arbiter.Activate(_taskQueue,
240         portDouble.Join(
241             portString, // port to join with
242             (value, stringValue) => // delegate
243             {
244                 value /= 2.0;
245                 stringValue = value.ToString();
246                 // post back updated values
247                 portDouble.Post(value + 1);
248                 portString.Post(stringValue + 1);
249                 Console.WriteLine("stringValue: " + stringValue);
250                 Console.WriteLine("value:" + value);
251             })
252         );
253
254     // post items. The order does not matter, which is what Join its power
255     portDouble.Post(3.14159);
256     portString.Post("0.1");
257
258     double item;
259     long totalSum = 0;
260     for (long i = 1; i <= 10000000; i++)
261         totalSum += i;
262     portDouble.Test(out item);
263     Console.WriteLine("portDouble Received item:" + item);
264
265     //after the last post the delegate above will execute
266 }
267
268 // create dispatcher and dispatcher queue for scheduling tasks
269 static Dispatcher dispatcher = new Dispatcher();
270 static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);
271 }
272 }

```

273 Example 11: Joint Race

```

274 using System;
275 using System.Collections.Generic;
276 using System.Text;
277 using Microsoft.Ccr.Core;
278
279 namespace Example
280 {
281     class Program
282     {
283         // Example 11: Joint Race

```

```

284     static void Main(string[] args)
285     {
286         /// <summary>
287         /// Using Joint to listen the same port will be a race, It cant guarentee that the Arbiter
288 will run.
289         /// </summary>
290
291         var portInt = new Port<int>();
292         var portDouble = new Port<double>();
293         var portString = new Port<string>();
294
295         // activate a joined receiver that will execute only when one
296         // item is available in each port.
297         Arbiter.Activate(_taskQueue,
298             portDouble.Join(
299                 portString, // second port to listen
300                 (value, stringValue) =>
301                 {
302                     value /= 2.0;
303                     stringValue = value.ToString();
304                     // post back updated values
305                     portDouble.Post(value);
306                     portString.Post(stringValue);
307                     Console.WriteLine("First stringValue:" + stringValue);
308                     Console.WriteLine("First value:" + value);
309                 })
310             );
311
312         // activate a second joined receiver that also listens on portDouble
313         // and on a new port, portInt. Because the two joins share a common port
314         // between them (portDouble), there is contention when items are posted on
315         // that port
316         Arbiter.Activate(_taskQueue,
317             portDouble.Join(
318                 portInt, // second port to listen
319                 (value, intValue) =>
320                 {
321                     value /= 2.0;
322                     intValue = (int)value;
323                     // post back updated values
324                     portDouble.Post(value);
325                     portInt.Post(intValue);
326                     Console.WriteLine("Second intValue:" + intValue);
327                     Console.WriteLine("Second value:" + value);
328                 })
329             );
330
331         // post items.
332         portString.Post("Test");
333         portInt.Post(128);
334
335         // when the double is posted there will be a race

```

```

336         // between the two joins to determine who will execute first
337         // The delegate that executes first will then post back a double,
338         // allowing the delegate that "lost", to execute.
339         portDouble.Post(3.14159);
340         //portString.Post("Test");
341     }
342
343     // create dispatcher and dispatcher queue for scheduling tasks
344     static Dispatcher dispatcher = new Dispatcher();
345     static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);
346 }
347 }

```

348 Example 12: Joint Wait

```

349 using System;
350 using System.Collections.Generic;
351 using System.Text;
352 using Microsoft.Ccr.Core;
353
354 namespace Example
355 {
356     class Program
357     {
358         // Example 12: Joint Wait
359         static void Main(string[] args)
360         {
361             /// <summary>
362             /// 全員到齊才會跑
363             /// </summary>
364
365             int itemCount = 10;
366             var portDouble = new Port<double>();
367
368             // post N items to a port
369             for (int i = 0; i < itemCount; i++)
370             {
371                 portDouble.Post(i * 3.14159);
372                 long totalSum = 0;
373                 for (long j = 1; j <= 100000000; j++)
374                     totalSum += j;
375                 Console.WriteLine("i: " + i);
376             }
377
378             // activate a Join that
379             // waits for N items on the same port
380             Arbiter.Activate(_taskQueue,
381                 portDouble.Join(
382                     itemCount, // wait itemCount times

```



```

383         items =>
384         {
385             foreach (double d in items)
386             {
387                 Console.WriteLine(d);
388             }
389         }
390     )
391 );
392 }
393
394 // create dispatcher and dispatcher queue for scheduling tasks
395 static Dispatcher dispatcher = new Dispatcher();
396 static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);
397 }
398 }

```

399 Example 13: Multiple Item receivers

```

400 using System;
401 using System.Collections.Generic;
402 using System.Text;
403 using Microsoft.Ccr.Core;
404
405 namespace Example
406 {
407     class Program
408     {
409         // Example 13: Multiple Item receivers
410         static void Main(string[] args)
411         {
412             /// <summary>
413             /// 全員到齊才會跑
414             /// </summary>
415
416             // create a simple service listening on a port
417             var servicePort = SimpleService.Create(_taskQueue); // 呼叫「ServicePort Create」
418
419             // shared response port
420             var responsePort = new PortSet<string, Exception>();
421
422             // number of requests
423             int requestCount = 10;
424
425             // scatter phase: Send N requests as fast as possible
426             for (int i = 0; i < requestCount; i++)
427             {
428                 long totalSum = 0;
429                 for (long j = 1; j <= 100000000; j++)

```

```

430         totalSum += j;
431         Console.WriteLine("i: " + i);
432
433         // Update the state of SimpleService
434         string item;
435         UpdateState _state = new UpdateState(); // public string State;
436         _state.State = "Test";
437         if (i > 6)
438             servicePort.Post(_state); // success
439
440         // create request
441         GetState get = new GetState();
442
443         // set response port to shared port
444         get.ResponsePort = responsePort;
445
446         // post request
447         servicePort.Post(get);
448     }
449
450     // gather phase:
451     // activate a multiple item receiver that waits for a total
452     // of N responses, across the ports in the PortSet.
453     // The service could respond with K failures and M successes (K+M == N)
454     Arbiter.Activate(_taskQueue,
455         responsePort.MultipleItemReceive(
456             requestCount, // wait requestCount times
457             (successes, failures) => Console.WriteLine("Total received (s vs f):" +
458 successes.Count + " " + failures.Count)
459         )
460     );
461 }
462
463 // create dispatcher and dispatcher queue for scheduling tasks
464 static Dispatcher dispatcher = new Dispatcher();
465 static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);
466
467 #region Example 13: function
468 /// <summary>
469 /// Example 13
470 /// </summary>
471 public class ServiceOperation
472 {
473     public PortSet<string, Exception> ResponsePort = new PortSet<string, Exception>(); // 建立ResponsePort的PortSet
474 }
475
476
477 public class Stop : ServiceOperation
478 {
479 }
480
481 public class UpdateState : ServiceOperation

```

```

482     {
483         public string State;
484     }
485
486     public class GetState : ServiceOperation
487     {
488     }
489
490     /// <summary>
491     /// PortSet that defines which messages the services listens to
492     /// </summary>
493     public class ServicePort : PortSet<Stop, UpdateState, GetState>
494     {
495     }
496     /// <summary>
497     /// Simple example of a CCR component that uses a PortSet to abstract
498     /// its API for message passing
499     /// </summary>
500     public class SimpleService
501     {
502         ServicePort _mainPort;
503         DispatcherQueue _taskQueue;
504         string _state;
505
506         public static ServicePort Create(DispatcherQueue taskQueue)
507         {
508             var service = new SimpleService(taskQueue);
509             service.Initialize();
510             return service._mainPort;
511         }
512
513         private void Initialize()
514         {
515             // using the supplied taskQueue for scheduling, activate two
516             // persisted receivers, that will run concurrently to each other,
517             // one for each item type
518             Arbiter.Activate(_taskQueue,
519                 Arbiter.Receive<UpdateState>(true, _mainPort, UpdateHandler),
520                 Arbiter.Receive<GetState>(true, _mainPort, GetStateHandler)
521             );
522         }
523
524         private SimpleService(DispatcherQueue taskQueue)
525         {
526             // create PortSet instance used by external callers to post items
527             _mainPort = new ServicePort();
528
529             // cache dispatcher queue used to schedule tasks
530             _taskQueue = taskQueue;
531         }
532
533         void GetStateHandler(GetState get)

```

```

534     {
535         if (_state == null)
536         {
537             // To demonstrate a failure response,
538             // when state is null will post an exception
539             get.ResponsePort.Post(new InvalidOperationException()); //
540             InvalidOperationException 內建錯誤回傳函式
541             return;
542         }
543
544         // return the state as a message on the response port
545         get.ResponsePort.Post(_state);
546     }
547     void UpdateHandler(UpdateState update)
548     {
549         // update state from field in the message
550         _state = update.State;
551
552         // as success result, post the state itself
553         update.ResponsePort.Post(_state);
554     }
555 }
556 #endregion
557 }
558 }

```