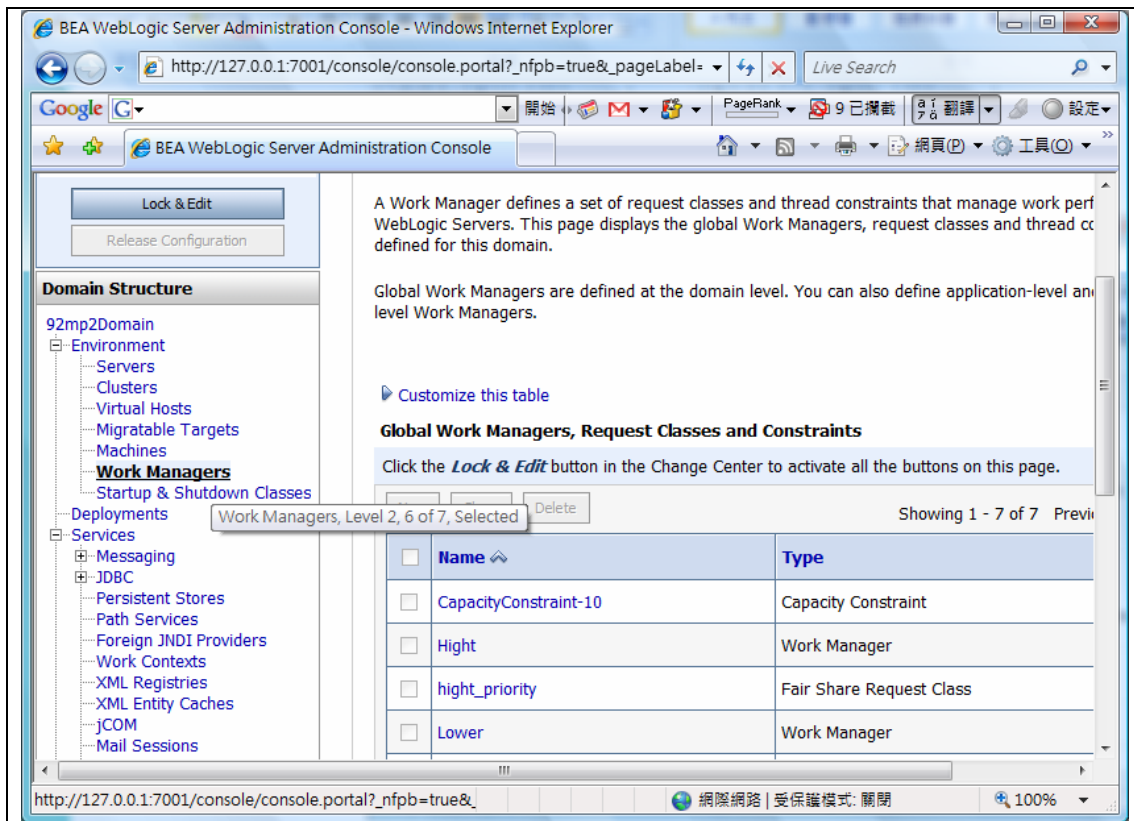


如何運用 Work Managers 提升系統效能

什麼是 Work Managers?

在 WebLogic Server 9.x 版本之前，在執行緒(thread)上的管理必需手動調整 thread pool 的數量，或建立多個 execute queues 來分散工作量，管理者需設定 thread count, thread priority 等參數。然而很多管理者常會問到的一個問題：到底我的 thread pool 的數量要設多少呢？

WebLogic Server 9.x 版在管理執行緒(thread)上為自我調校(self-tuning)，換句話說就是 WebLogic Server 會動態地增加或減少 thread pool 的數量。什麼是 Self-tuning 呢？它的意思是 WebLogic Server 建立了一個單一的 thread pool，所有的工作都會使用到這個 thread pool，並且 server 會監視整個工作使用的情形，自動動態地調整 thread pool 的數量，來增加系統效能。WebLogic Server 9.x 版也提供管道讓管理者可以自行調整執行緒(thread)，稱為 Work Managers。下圖為 Work Managers 的管理介面。



Default Work Manager

WebLogic Server 會有一個 default 的 Work Manager，所有的 Application 都有相等的執行權(fair share)，假如沒有特別指定 Work Manager 給 Application，它會使用 default 的 Work Manager。管理者可以建立一個名稱為「default」的 Work Manager 來覆蓋原有的 default Work Manager。

March 05 M-Power eNew

本篇文章版權為倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

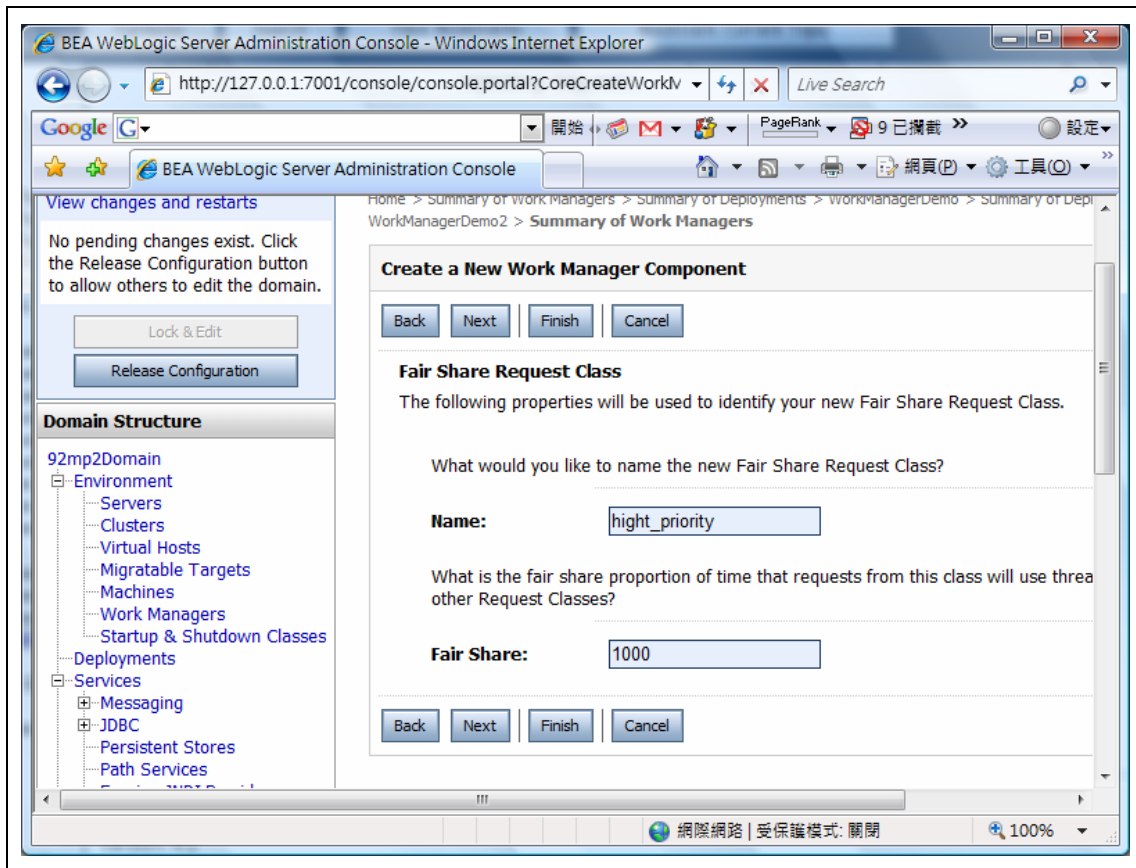
Work Manager 的元件

Request Classes

此元件可以用來設定工作執行的優先順序，此元件共有三種不同的型態：
fair-share-request-class, response-time-request-class, context-request-class。

WebLogic Server 會依據設定的條件來執行工作。

fair-share-request-class 用來指定 thread-usage time，default 值為 50，最小值為 1，最大值為 1000。假設你有一個需要高優先執行的 Application，可以將其 fair-share-request-class 的值設為 1000，如下圖所示。



response-time-request-class 則以 response time 為目標，將其與平均 thread 的使用時間相減來求取可容許之等待時間，並讓此類 request 的平均等待時間與可容許之等待時間成比例，至於 response time 的單位為 millisecond，default 值為 0。

context-request-class 則是以 user 或是 group 來指定為基準來指定其所參照到的 request class 類別。

要注意的是設定 Request Classes 只能為上述三種型態中其中一種，並不能同時設定兩種以上的 Request Classes。

Constraints

此元件包含 max-threads-constraint , min-threads-constraint , capacity constraint 。

max-threads-constraint 用來指定最大同時可使用執行緒(concurrent threads)數量，default 值為 -1 。

min-threads-constraint 用來指定保證最小可同時使用的執行緒數量(concurrent threads)，default 值為 1 。

capacity constraint 用來設定當 pending request 的數量到達時，WebLogic Server 會自動拒絕 request，避免 server deadlock，default 值為-1 。

在配置 Work Manager 時必需定義至少有 Request Classes 或是 Constraints 其中的一種，也可以同時定義兩種，但不能同時定義兩種的 Request Classes 或是兩種的 Constraints 。

配置 Work Managers

Work manager 有四個層級的組態方式

Global 層級

顧名思義就是將 Work Manager 組態於 domain 中，就是將內容儲存於 config.xml 並且 target 到某些特定的 server instance 上，這種設定方式可以讓所有部署在該 server 上的 application 或是 module 參照到。

Application 層級

就是在佈署 Ear 模組時，將要使用的 Work Manager 定義在 weblogic-application.xml 檔中，使用這種方式只有該 application 或其 module 可以參照到該 Work Manager 。

```
<weblogic-application xmlns="http://www.bea.com/ns/weblogic/90"
  xmlns:j2ee="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bea.com/ns/weblogic/90
  http://www.bea.com/ns/weblogic/90/weblogic-application.xsd">

  <max-threads-constraint>
    <name>j2ee_maxthreads</name>
    <count>1</count>
  </max-threads-constraint>

  <min-threads-constraint>
    <name>j2ee_minthreads</name>
    <count>1</count>
  </min-threads-constraint>

  <work-manager>
    <name>J2EEScopedWorkManager</name>
  </work-manager>
</weblogic-application>
```

March 05 M-Power eNew

本篇文章版權為倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

Component 層級

這裡只的 component 層級又叫做 ejb 層級，以就是將 Work Manager 定義在 weblogic-ejb-jar.xml 檔中，供這裡的 ejb 元件參照用。

```
<weblogic-ejb-jar xmlns="http://www.bea.com/ns/weblogic/90"
  xmlns:j2ee="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bea.com/ns/weblogic/90
  http://www.bea.com/ns/weblogic/90/weblogic-ejb-jar.xsd">

  <weblogic-enterprise-bean>
    <ejb-name>ResourceConstraintEJB</ejb-name>
    <jndi-name>core_work_ejb_resource_ResourceConstraintEJB</jndi-name>
    <dispatch-policy>test_resource</dispatch-policy>
  </weblogic-enterprise-bean>

  <weblogic-enterprise-bean>
    <ejb-name>AppScopedResourceConstraintEJB</ejb-name>
    <jndi-name>core_work_ejb_resource_AppScopedResourceConstraintEJB
    </jndi-name>
    <dispatch-policy>test_appscoped_resource</dispatch-policy>
  </weblogic-enterprise-bean>

  <work-manager>
    <name>test_resource</name>
    <max-threads-constraint>
      <name>pool_constraint</name>
      <pool-name>testPool</pool-name>
    </max-threads-constraint>
  </work-manager>

  <work-manager>
    <name>test_appscoped_resource</name>
    <max-threads-constraint>
      <name>appscoped_pool_constraint</name>
      <pool-name>AppScopedDataSource</pool-name>
    </max-threads-constraint>
  </work-manager>
</weblogic-ejb-jar>
```

Web Application 層級

將 Work Manager 定義在 weblogic.xml 中，供特定的 web module 呼叫。

```
<?xml version="1.0" encoding="UTF-8"?>

<weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90"
xmlns:j2ee="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/ns/weblogic/90
http://www.bea.com/ns/weblogic/90/weblogic-web-app.xsd">

  <work-manager>
    <name>foo-servlet-1</name>
    <request-class-name>test-fairshare2</request-class-name>
    <max-threads-constraint>
      <name>foo-mtc</name>
      <pool-name>oraclePool</pool-name>
    </max-threads-constraint>
  </work-manager>

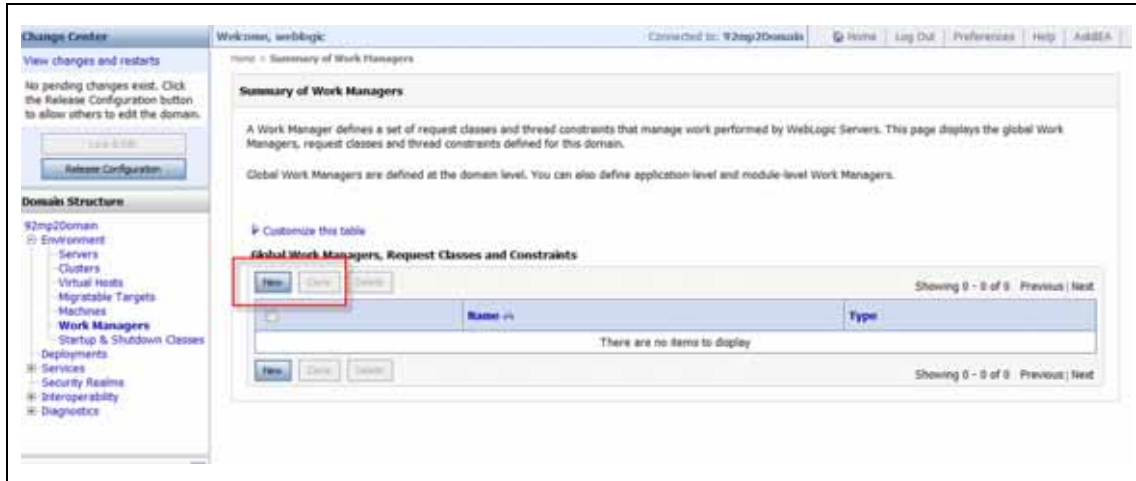
  <work-manager>
    <name>foo-servlet</name>
    <context-request-class>
      <name>test-context</name>
      <context-case>
        <user-name>anonymous</user-name>
        <request-class-name>test-fairshare1</request-class-name>
      </context-case>

      <context-case>
        <group-name>everyone</group-name>
        <request-class-name>test-fairshare2</request-class-name>
      </context-case>
    </context-request-class>
  </work-manager>
</weblogic-web-app>
```

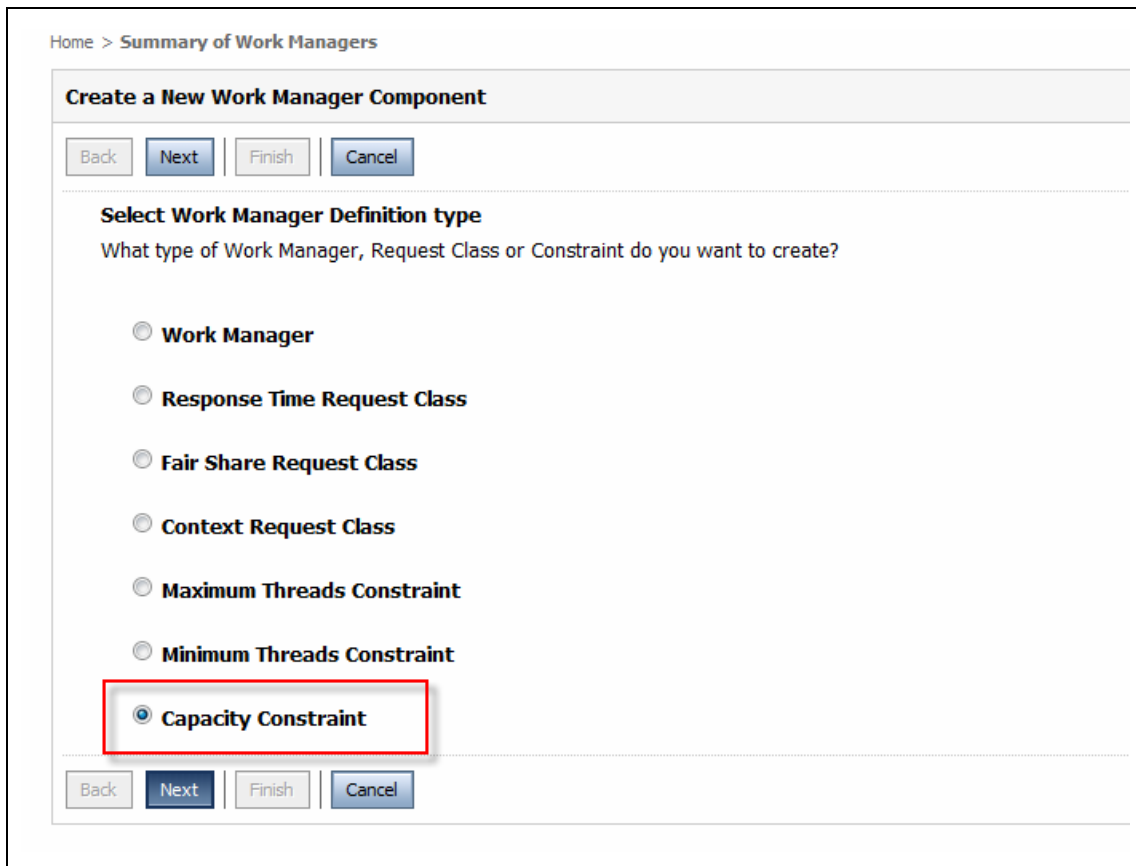
範例解說

目前單就 Global 層級的設定方式做介紹，我們可使用 max-threads-constraint 與 capacity 混合使用的範例來做講解。

在 Admin Console 的左邊 menu bar 上 locate 到 <Domain>/Environment/Work Managers，點選 lock and edit 後按下 New 新增一個 Work Manager。



選擇 capacity constraint 後點選 Next。



替這個 constraint type 取得名字與設定其 count 數，count 設預設值-1 代表的是無上限，我們為測試的目的可以將其設定為 10，之後點按 Next。

Home > Summary of Work Managers

Create a New Work Manager Component

Back Next Finish Cancel

Capacity Constraint Properties

The following properties will be used to identify your new Capacity Constraint.

What would you like to name the new Capacity Constraint?

Name:

How many requests should be queued or running before WebLogic Server begins rejecting requests?

Count:

Back Next Finish Cancel

將其 target 到某個 server instance，爲了測試的目的用，將其 target 到 AdminServer 後，按下 Finished。

Home > Summary of Work Managers

Create a New Work Manager Component

Back Next Finish Cancel

Select deployment targets

You can target the Work Manager to any of these WebLogic Server instances or Clusters. Select to reference the Work Manager.

Available targets

Servers	
<input checked="" type="checkbox"/>	AdminServer

Back Next Finish Cancel

按照之前的步驟在建一個新的 Constraint，這次選 Maximum Threads Constraint，並點按 Next。

Home > Summary of Work Managers

Create a New Work Manager Component

Back Next Finish Cancel

Select Work Manager Definition type
What type of Work Manager, Request Class or Constraint do you want to create?

- Work Manager
- Response Time Request Class
- Fair Share Request Class
- Context Request Class
- Maximum Threads Constraint**
- Minimum Threads Constraint
- Capacity Constraint

Back Next Finish Cancel

填寫 name 與 count，count 預設值為-1 表示無上線，為了測試目的將其改為 7，之後按下 Next 並將其 target 到 AdminServer，最後按下 Finished。

Home > Summary of Work Managers

Create a New Work Manager Component

Back Next Finish Cancel

Maximum Threads Constraint Properties
The following properties will be used to identify your new MaxThreads Request Class.

What would you like to name the new Maximum Threads Constraint?

Name:

What is the maximum number of concurrent threads to allocate for requests? Enter either a fixed thread count or the name of a Data Source whose size will be used for the constraint.

Count:

Data Source:

Back Next Finish Cancel

最後執行的步驟是建立一個 Work Manager 並且將其上建置的兩個 Constraints 加入到 Work Manager。

Home > Summary of Work Managers

Create a New Work Manager Component

Back Next Finish Cancel

Select Work Manager Definition type
What type of Work Manager, Request Class or Constraint do you want to create?

Work Manager

Response Time Request Class

Fair Share Request Class

Context Request Class

Maximum Threads Constraint

Minimum Threads Constraint

Capacity Constraint

Back Next Finish Cancel

將其命名為 wm1 後點按 Next。

Home > Summary of Work Managers

Create a New Work Manager Component

Back Next Finish Cancel

Work Manager Properties
The following properties will be used to identify your new Work Manager.

What would you like to name your new Work Manager?

Name:

Back Next Finish Cancel

將其 target 到 AdminServer。

在 Work Manager 的 summary list 上點選 wm1，在 configuration 下將之前 config 的 maxT1 與 cc1 這兩個 Constraints 加入到 wm1，並點選 Save。

最後一個步驟將將之前的設定生效，點按 Active Changes。

註: Constraints 與 Request Class 也可以個別被 application 或 module 所參照，不一定要包在 Work Manager 下。

再來我們可以開始撰寫測試用的程式碼來印證我們之前的設定，我們可以先寫一個 Servlet 程式來模擬處理 Client 的 request，程式碼的片段如下，在這裡我們使用 Thread 的 sleep 方法來模擬 Thread 正在處理大量費時工作，延遲 Thread 交還給 Thread Pool 的時間。

```
/* (non-Java-doc)
 * @see javax.servlet.http.HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/plain");
    response.getWriter().println("servlet has been requested !!");
    try {
        Thread.sleep(60000L);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

除了程式碼之外，另外一個最重要一個 config 就是開發人員須在 weblogic.xml 檔中將該 web module 參照到之前設定的 Work Manager，這樣才可以使 Work Manager 的相關設定套用到該 web module。

```
1<?xml version="1.0" encoding="UTF-8"?>
2<wls:weblogic-web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:wls="http://www.bea.com/
3  <wls:context-root>WMTTest</wls:context-root>
4  <wls:wl-dispatch-policy>wml</wls:wl-dispatch-policy>
5</wls:weblogic-web-app>
```

之後將 web module 打包後部屬在 weblogic server 上。

接下來我們需要準備的是寫一個測試用的程式，我們可以利用 java 提供的 Thread 物件來啓用多個執行緒模擬多個使用者同時對 Server 做 request 的動作。

片段一(注意要 implement Runnable 介面)。

```
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class WMClient implements Runnable {

    public void run() {
        HttpURLConnection conn = null;
        InputStream in = null;
        try {
            URL url = new URL("http://localhost:7001/WMTest/TestServlet");
            conn = (HttpURLConnection)url.openConnection();
            conn.setReadTimeout(65000);
            conn.connect();
            int status = conn.getResponseCode();
            System.out.println("response code: " + status);
            if (status == 200) {
                in = conn.getInputStream();
                byte[] b = new byte[1024];
                int count = in.read(b);
                System.out.println("response message: " + new String(b).trim());
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if(in != null) {
                try {
                    in.close();
                } catch (IOException e) {}
            }
        }
    }
}
```

片段二(設定 15 個 Thread 代表同時有 15 個使用者做 request)。

```
/**
 * @param args
 */
public static void main(String[] args) {
    Thread[] ts = new Thread[15];
    for(Thread t: ts) {
        t = new Thread(new WMClient());
        t.start();
    }
}
```

執行程式後可以發現幾個結果:

超過之前設定的 capacity 數 10 的部份立刻被 reject 了(HTTP Status Code 為 503)。

其中 7 個 request 由於沒有超過之前設定的 max-threads-constraint 以及在 Client 程式的 Reading Timeout 時線內回覆所以 HTTP Status Code 均為 200，且順利讀取 Server 回覆的內容。

有 3 個 exception 為 Reading Timeout 的錯誤發生。

March 05 M-Power eNew

本篇文章版權為倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

```
response code: 503
response code: 503
response code: 503
response code: 503
response code: 503
response code: 200
response code: 200
response code: 200
response code: 200
response code: 200
response code: 200
response code: 200
response message: servlet has been requested !!
response message: servlet has been requested !!
response message: servlet has been requested !!
response message: servlet has been requested !!
response message: servlet has been requested !!
response message: servlet has been requested !!
java.net.SocketTimeoutException: Read timed out
    at jrockit.net.SocketNativeIO.readBytesPinned(Native Method)
    at jrockit.net.SocketNativeIO.socketRead(Unknown Source)
    at java.net.SocketInputStream.socketRead0(SocketInputStream.java)
    at java.net.SocketInputStream.read(SocketInputStream.java:129)
    at java.io.BufferedInputStream.fill(BufferedInputStream.java:218)
    at java.io.BufferedInputStream.read1(BufferedInputStream.java:256)
    at java.io.BufferedInputStream.read(BufferedInputStream.java:313)
    at sun.net.www.http.HttpClient.parseHTTPHeader(HttpClient.java:626)
    at sun.net.www.http.HttpClient.parseHTTP(HttpClient.java:571)
    at sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:961)
    at java.net.HttpURLConnection.getResponseCode(HttpURLConnection.java:367)
    at test.WMClient.run(WMClient.java:18)
    at java.lang.Thread.run(Thread.java:595)
java.net.SocketTimeoutException: Read timed out
    at jrockit.net.SocketNativeIO.readBytesPinned(Native Method)
    at jrockit.net.SocketNativeIO.socketRead(Unknown Source)
```

與舊版 WebLogic 的 Execution Queue 相容

WebLogic 9 亦有提供與舊版 WebLogic 的 Execution Queue 相容的設定，方法有二：

使用-D 命令列選項

-Dweblogic.Use81StyleExecuteQueues=true

變更 config.xml 檔

```
<server>
  <name>myserver</name>
  <ssl>
    <name>myserver</name>
    <enabled>true</enabled>
    <listen-port>7002</listen-port>
  </ssl>
  <use81-style-execute-queues>true</use81-style-execute-queues>
  <listen-address/>
</server>
```

執行變更後，我們之前所配置的 Work Manager 就會變成等同於 Execution Queue 的性值了，

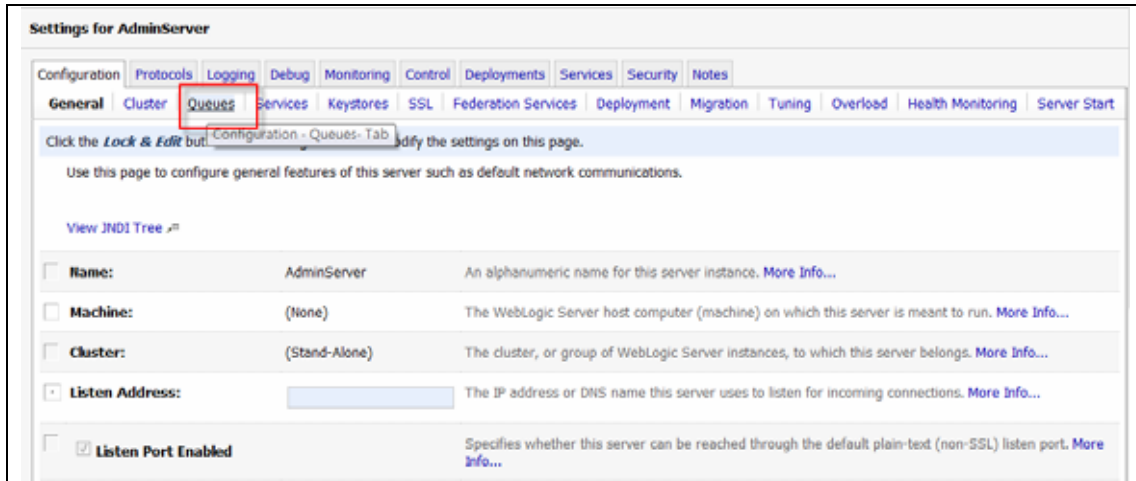
March 05 M-Power eNew

本篇文章版權為倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

在重新啓動 WebLogic 時可以注意預設的 self-tuning 被關掉了。

```
state changed to STARTING>  
<2008/2/29 下午02時54分53秒 CST> <Notice> <Kernel> <BEA-000805> <Self-tuning thread pool is disabled. An execute queue will be created for each WorkManager definition.>
```

另外 Server 的 Configuration 頁籤中多了 Queues 這個頁籤。



Work Manager 可視管理上的需求加以組態，例如當管理者認為某個應用程式的模組需要有較高的執行優先權，而既有的 Thread 分配方式不符合需求，此時即可組態 Work Manager 來符合使用上的優先順序，達成管理上的目標。

以上所列均為小部份的應用，筆者之舉例希望可以收到拋磚引玉的效果，至於更詳細的設定讀者可至 BEA 的官網查詢。