

## 運用 Instrumentation 協助程式效能偵測與除錯

### 何為 Instrumentation?

Weblogic 9.x 之後將對系統監控及診斷之能力抽象成爲一個 Framework，稱之爲 Diagnostic Framework，管理者可以使用他來建立、蒐集、分析 Server 與 Application 的執行時期資料，而其中一項功能名爲 Instrumentation，其功用爲以在不更動原來程式的方式，於執行時期增加診斷功能於服務或是程式碼之上。

Instrumentation 可以針對 Weblogic 系統層級或是應用程式層級進行診斷介入，本次的說明將會示範應用程式層級的設定。

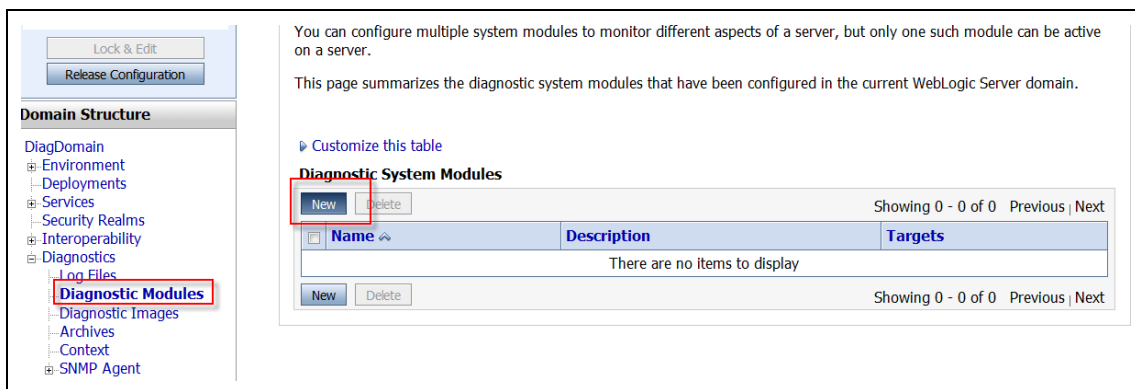
一般的程式開發人員在進行系統除錯時，如欲確定方法輸入的參數是否正確，又或者相要了解方法執行之速率，以協助確認執行效率瓶頸之所在，常習慣於方法內將執行時期輸入的參數或計算後方法的執行時間 Dump 出來，可能是個 System.out.println() 的方法呼叫或是使用 third-party 的 logging 套件來輸出 debug 資料。

然過多的除錯用程式碼容易造成程式的雜亂導致日後維護的不易，一般專案負責 code review 之資深工程師會要求其下工程師在測試或偵錯完畢後須將其移除，然移除與加入偵錯或測試程式碼均需要對原始碼改動，開發人員行之難免有錯，因此軟體界部分人士提出所謂的 AOP(Asspect Orient Programming) 概念，以物件的方法爲切點，在呼叫方法前或是方法後進行某些攔截進而介入行爲，例如，在呼叫方法前進行交易的宣告，在呼叫方法完成後進行交易確認的動作，對原本元件之開發者而言具備完整的通透性。

而 Weblogic 則利用 AOP 的概念將其套用在 Diagnostic Framework 中的 Instrumentation 功能之中，以下會有一個簡單的範例示範如何利用 Instrumentation 作效能之衡量。

### 前置作業

在使用 Instrumentation 前須先建置一個 Diagnostic Module 並且 target 到一個 Server Instance 上，請利用 Admin Console 左選單上 Diagnostics/Diagnostics Modules/New 建置。



The screenshot shows the WebLogic Admin Console interface. On the left, the 'Domain Structure' tree is visible, with 'Diagnostic Modules' highlighted under the 'Diagnostics' folder. The main content area displays a message: 'You can configure multiple system modules to monitor different aspects of a server, but only one such module can be active on a server. This page summarizes the diagnostic system modules that have been configured in the current WebLogic Server domain.' Below this, there is a 'Diagnostic System Modules' table with columns for 'Name', 'Description', and 'Targets'. The table is currently empty, showing 'Showing 0 - 0 of 0' items. A 'New' button is visible above the table, and another 'New' button is visible below the table.

**System Module Properties**

The following properties will be used to identify your new system module.

\* Indicates required fields

What would you like to name your new module?

**\*Name:**

Provide a description for your new module.

**Description:**

OK Cancel

**Settings for Module-0**

Configuration Targets

Save

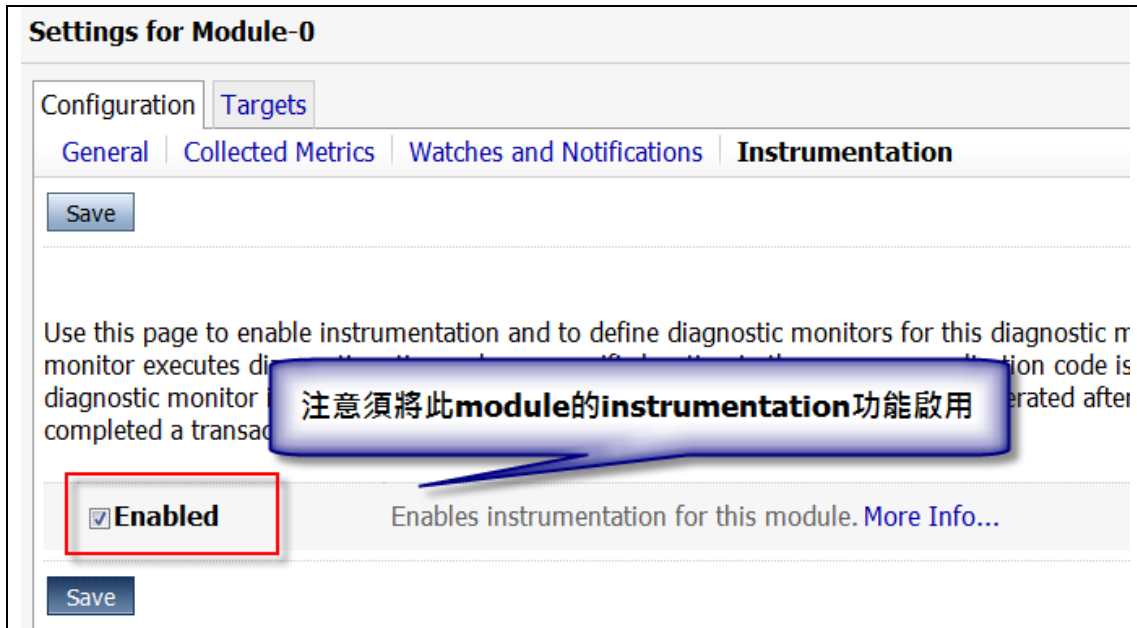
Use this page to specify the target settings for this diagnostic system module.

**Servers**

AdminServer

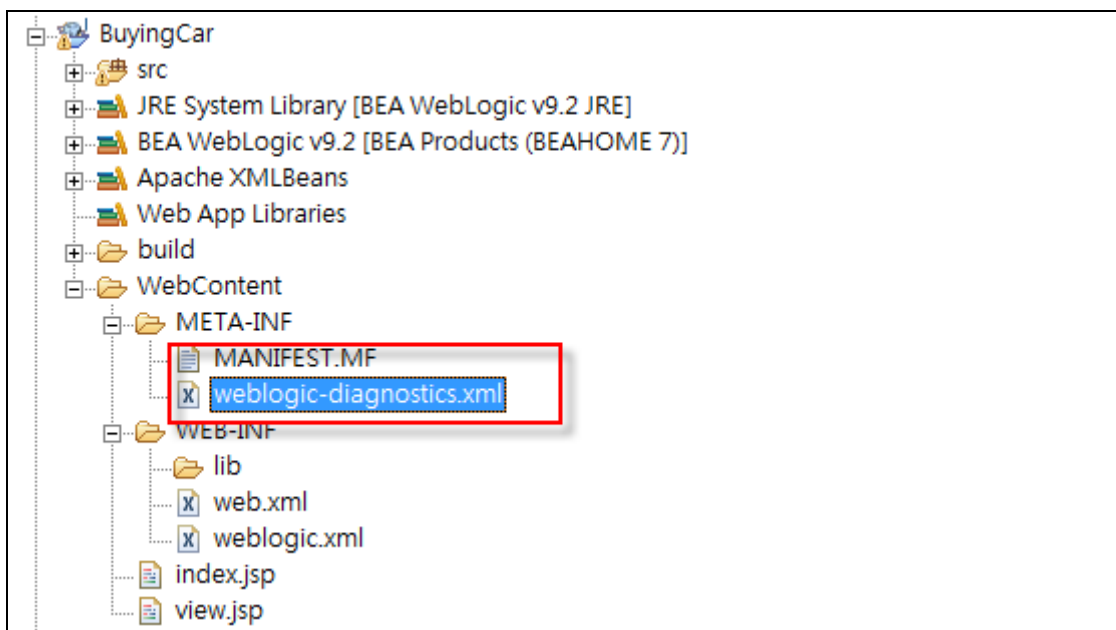
Save

需target到特定server上



## 利用 Instrumentation 檢查程式執行效率

本次將以一個 Web Application 為說明的範例，在符合 J2EE Spec 的應用程式模組中將 META-INF 資料匣中加入一個名為 weblogic-diagnostics.xml 的檔案。



再來要確定程式被呼叫的方法所輸入的參數是否正確並符合預期，可以利用 Instrumentation 的功能來預先定義一組 location-type 為 around 的 TraceElapsedTimeAction，並將其加入 xml 檔中，其意義為在呼叫該方法前先將方法輸入參數 log 出來，至於 match 方法的指令亦列示如下(語法為 AspectJ 的 pointcut 語法)：

```

1 <wldf-resource xmlns="http://www.be
2   xmlns:xsi="http://www.w3.org/20
3   xsi:schemaLocation="http://www
4
5   <instrumentation>
6     <enabled>true</enabled>
7     <wldf-instrumentation-monitor
8       <name>getItems</name>
9       <enabled>true</enabled>
10      <action>TraceElapsedTimeAction</action>
11      <location-type>around</location-type>
12      <pointcut>call(* test.ItemDao getItems(...))</pointcut>
13    </wldf-instrumentation-monitor>
14  </instrumentation>
15
16

```

攔截test.ItemDao的getItems方法

之後將此 Web Application 打包成爲一個 War 檔後，使用 Weblogic 應用程式部署介面將其部署爲一個 Web Application，爲求在 log 執行時間時有比較明顯的感覺，測試程式碼的部分有刻意用 Thread.sleep 方法將執行緒先暫停 5 秒。

```

57
58 public List getItems() {
59   try {
60     Thread.sleep(5000);
61   } catch (InterruptedException e1) {
62     e1.printStackTrace();
63   }
64   Connection conn = null;
65   PreparedStatement pstmt = null;
66   ResultSet rs = null;
67   List list = new ArrayList();
68   try {
69     conn = ds.getConnection();
70     pstmt = conn.prepareStatement("select * from items");
71     rs = pstmt.executeQuery();
72     while (rs.next()) {
73       Item item = new Item();
74       item.setName(rs.getString("name"));
75       item.setDesc(rs.getString("desc"));
76       item.setPrice(rs.getInt("price"));
77       list.add(item);
78     }
79   } catch (SQLException e) {
80     e.printStackTrace();
81   } finally {

```

在實際執行 Web Application 時會在呼叫到此物件的方法時有明顯的停頓，屆時可容易看出其差異。

## 檢視程式執行時間

在 Admin Console 的左選單上，選擇 Diagnostics/Log Filters 後，在右邊頁面上選擇 EventDataArchive，並選擇 View，可看到 TraceElapsedTimeAction 將 getItem 方法呼叫前與後之時間 log 下來，而其差異正是我們在程式中所刻意製造的 5 秒暫停。

Date ^	Type	Monitor	Class	Method	Payload
09/01/08 01:57:56 423	TraceElapsedTimeAction-Before-3	getItem	test.ItemDao	getItem	
09/01/08 01:58:01 424	TraceElapsedTimeAction-After-3	getItem	test.ItemDao	getItem	5001247016
09/01/08 01:58:01 424	TraceElapsedTimeAction-Before-4	getItem	test.ItemDao	getItem	
09/01/08 01:58:06 425	TraceElapsedTimeAction-After-4	getItem	test.ItemDao	getItem	5000894877

## 結語

筆者在此示範的方式為直接變更 weblogic-diagnostics.xml 檔，然實務上亦可以利用 Admin Console 直接定義 deploy plan 來動態的覆蓋 weblogic-diagnostics.xml 的結果。

筆者在範例 weblogic-diagnostics.xml 所列出的的 action 與 location-type 均為定義中之一小部分，若讀者有心深研，建議可參考 Oracle Bea 官方網站中有關 Instrumentation 的部分 ([http://e-docs.bea.com/wls/docs92/wldf\\_configuring/config\\_instrumentation.html](http://e-docs.bea.com/wls/docs92/wldf_configuring/config_instrumentation.html))。