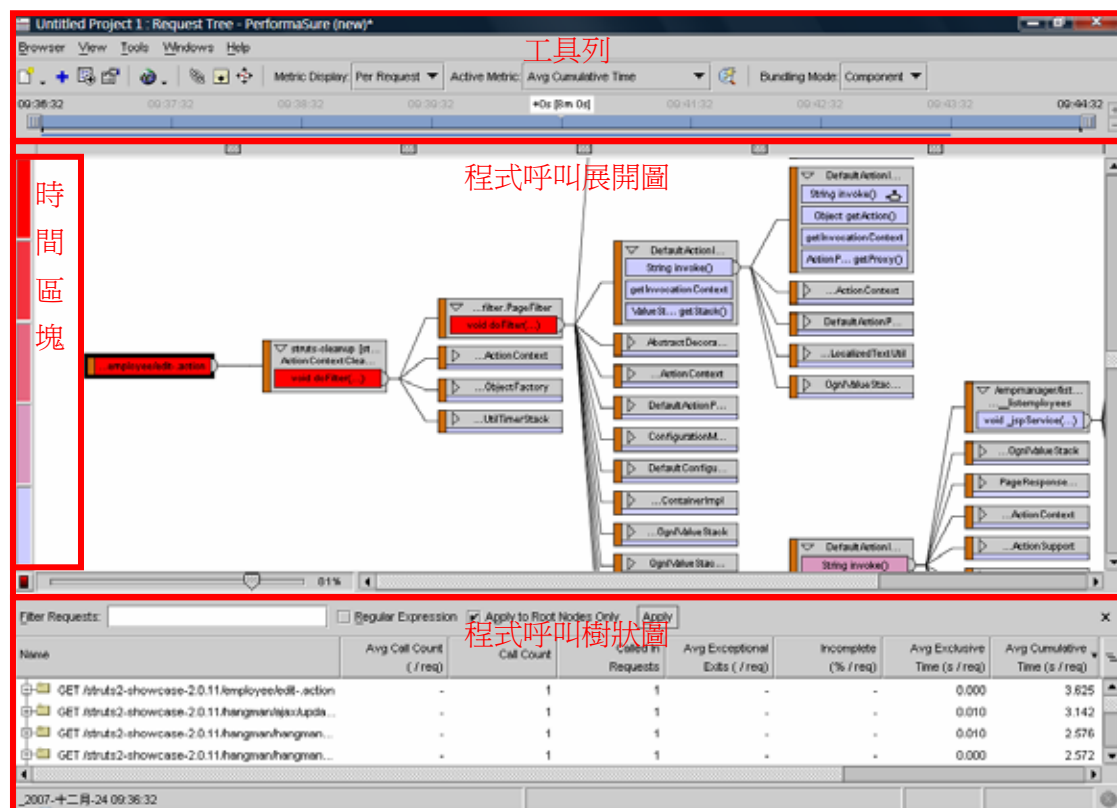


如何快速找到效能瓶頸點？

當我們使用 PerformaSure 來做效能瓶頸分析時，在分析資料時我們最常會使用 request-tree 這個瀏覽介面進行 Overall 的效能分析，除了可以透過下面的資料列表中對特定的欄位做排序外，還可以善用其所提供的快速鍵與篩選器，讓我們能快速找到瓶頸點。

Request Tree Browser

Request Tree Browser 為顯示應用程式的呼叫流程，它是可以橫跨在被監看的 Application Server，它的畫面如下：



圖(一) Request Tree Browser

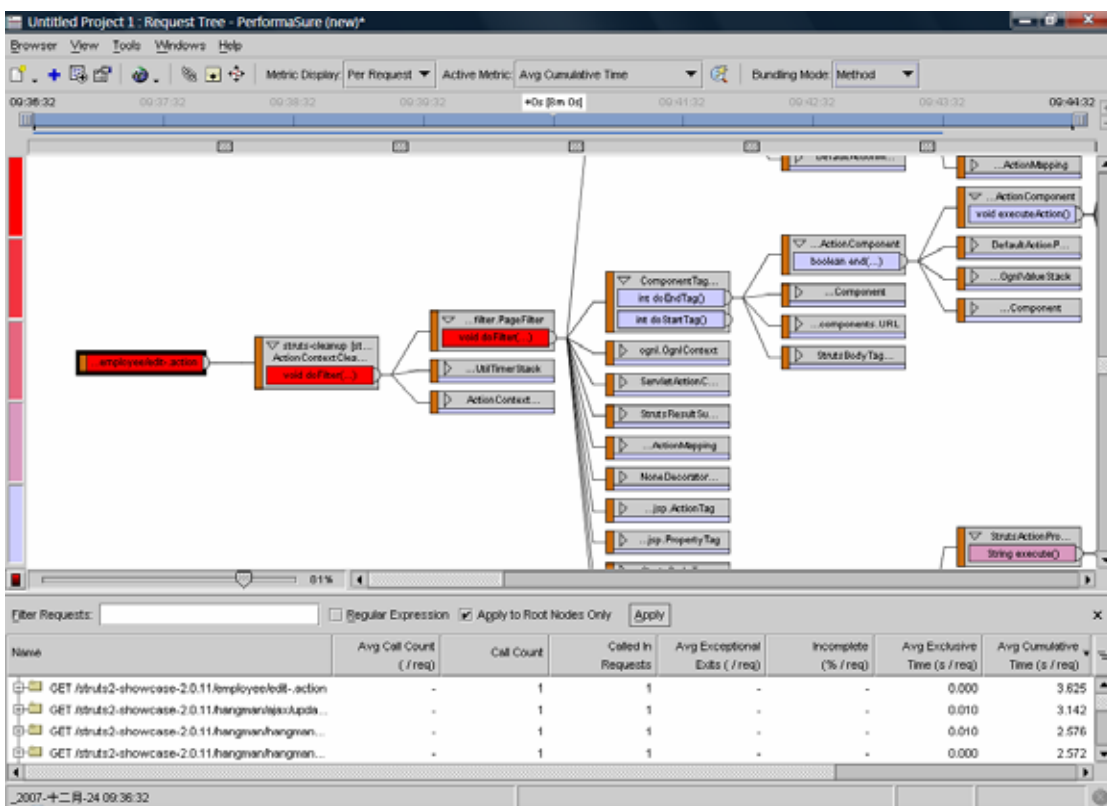
圖(一)為 Request Tree Browser 一打開時的畫面，它可以分為上下三個部份，最上面是工具列，中間則是整個程式呼叫展開圖，而最下方則為程式呼叫的樹狀圖，而在展開圖的最左方有一條顏色區塊，這個顏色區塊是代表展開圖中，所統計出來的時間區塊，因此在展開圖中每一個方塊最左邊的顏色則便是以時間區塊來呈現所耗的時間，以讓我們能快速在展開圖中找到瓶頸的物件。

當我們一打開時 Request Tree Browser 時中間的展開圖預設是以 Component 來做效能的呈現，但就要分析應用程式瓶頸的話，以 component 的角度對於分析是會有茫點的，就系統執行的邏輯是在物件的 Method 裡面，所以我們若要監看效能瓶頸則必需以 Method 較為準確，因

此我們可以善用篩選工具(Bundling Mode)來調整所呈現的資料，將篩選工具的選項由預設的 Component 改為 Method(圖二)，這樣便能完整呈現出整個執行過程中的程式呼叫，調整後便會呈現以 Method 角度的展開圖(圖三)。

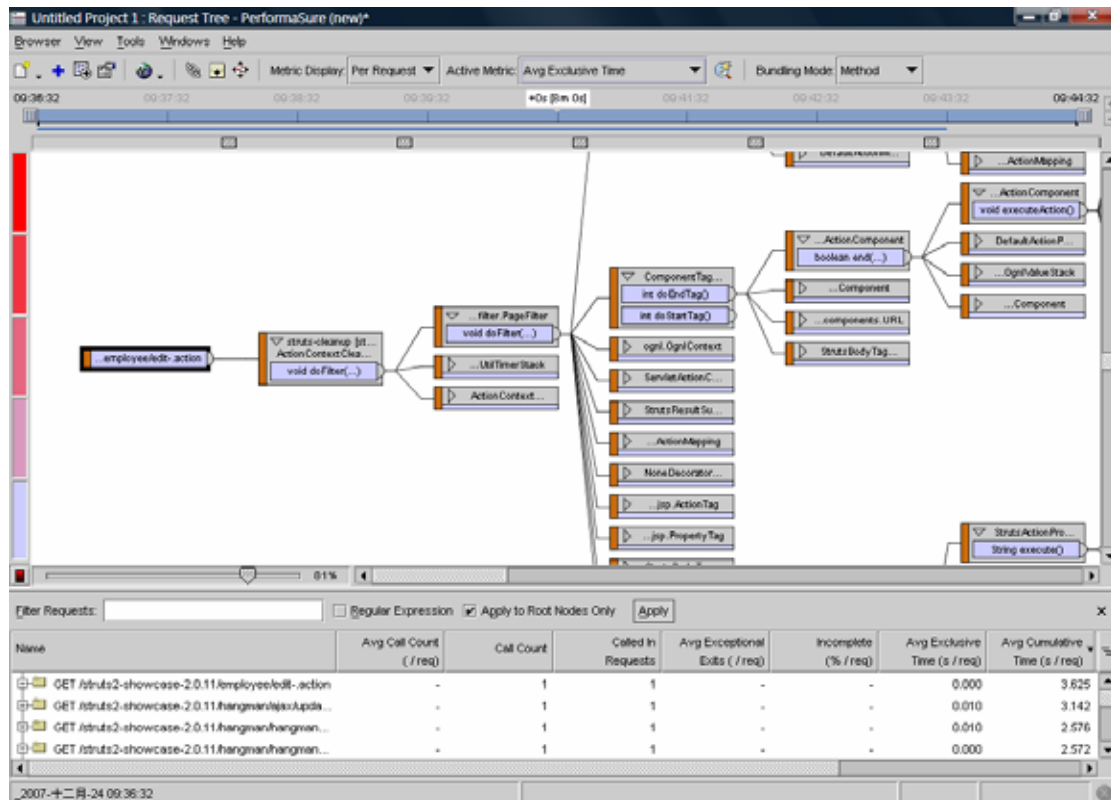


圖(二) 調整篩選工具



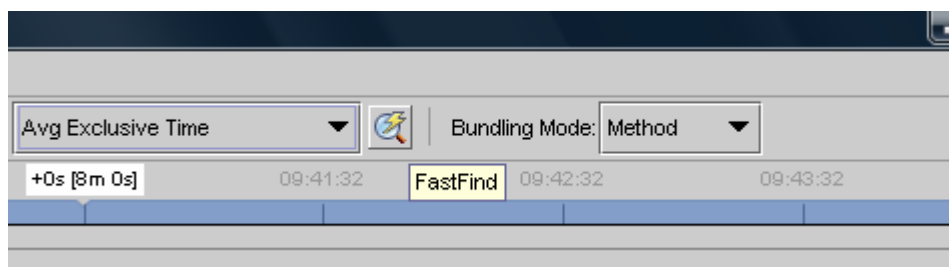
圖(三) Request Tree Browser(二)

當我們調整好 Request Tree Browser 之後，再來我們便可以開始來找尋應用系統效能瓶頸，我們要找效能瓶頸點則要知道是那一個元件的方法(Method)執行過慢，因此我們要調整資料統計條件(Active Metric)，由預設的 Avg(Average) Cumulative Time 改為 Avg(Average) Exclusive Time(圖四)，這個統計條件仍是採用每一個方法(Method)在監看的過程中所執行的加總平均時間，因此它所呈現的資訊便能讓我們可方便找到問題點。

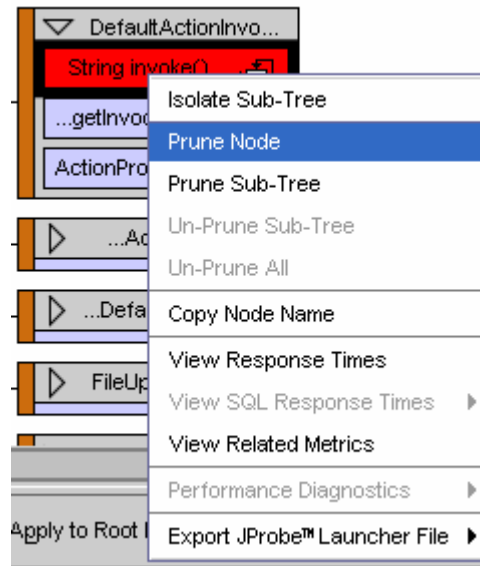


圖(四) Request Tree Browser(三)

再來我們要找到效能瓶頸點則可以透過工具列上面的快速鍵(FastFind 圖五)，來找目前效能最差的方法，若找到目前效能最佳的方法後，若發現這個 Method 後面還有其他的方法，則可以確定問題點就是它；若還想再繼續找其他效能不佳的方法，則可以將剛剛所找到的方法先做註記(Prune Node，圖六)，然後再透過快速鍵再找尋下一個效能不佳的方法；透過這樣反覆的操作，將所有效能不佳的方法一一找出來，再從中一一去看這些方法中到底是在做那些執行程式，找到真正的問題點便不遠了。



圖(五) 快速鍵(FastFind)



圖(六) 註記(Prune Node)