

使用 Annotation 簡單開發 EJB 3.0

前言

Java EE 的應用程式開發者，皆知有 EJB 這種元件，但很長的一段時間，對多數開發者而言，他只是個“名詞”，此話怎講，在 EJB3.0 規格出來之前，要撰寫 EJB 真是談何容易，要開發個可用的元件出來，還得先了解到 EJB 提供的 N 個 interface，Home Interface、Remote Interface、EJBObject、又有真正負責商業邏輯的 EJB Bean 本身，其中的設定，牽一髮而動全身，搞的開發人員唉唉叫，最後的結果就是，一大掛的開發人員通通投到了 Spring Framework 的懷抱中了，因為人家只要搞一個 Interface，一個 Implementation，就好了，還可以享盡舊有 EJB 關於宣告式的 Security 機制的好處。

EJB 的規格制定者們一看不得了，決定針對 EJB 的設定方式與 Interface 的規定進行大改，所謂的設定方式改採 Java 1.5 時所引入的 Annotation，interface 的部份，則從善如流，乾脆整個打掉不要；而另一個 EJB 的概念就是針對其中一種 EJB，Entity Bean，讓這種 Bean 可以不需要 EJB Container 即可運行，此概念深受當紅 ORM Framework Hibernate 的影響，這就不在本文討論的範圍了。

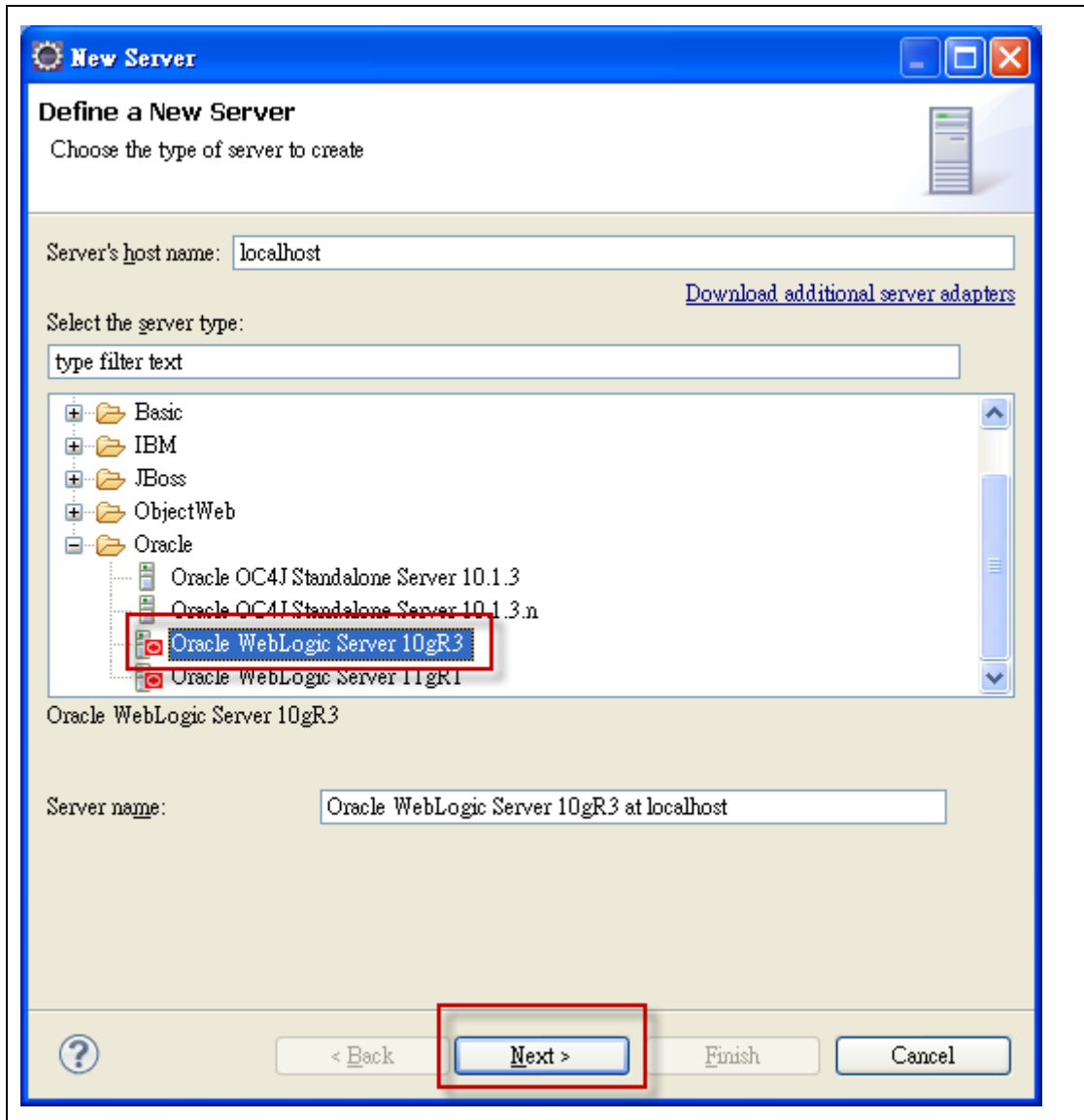
至於 EJB 的 Annotation，你可以將它想像成一組描述 EJB 程式的標籤即可，一旦程式加上了這種描述的標籤，EJB Container 就會知道它將來是個 EJB，它的名字叫甚麼，他的哪幾個程式的方法是允許別人呼叫的，聽起來挺神奇的對吧!!

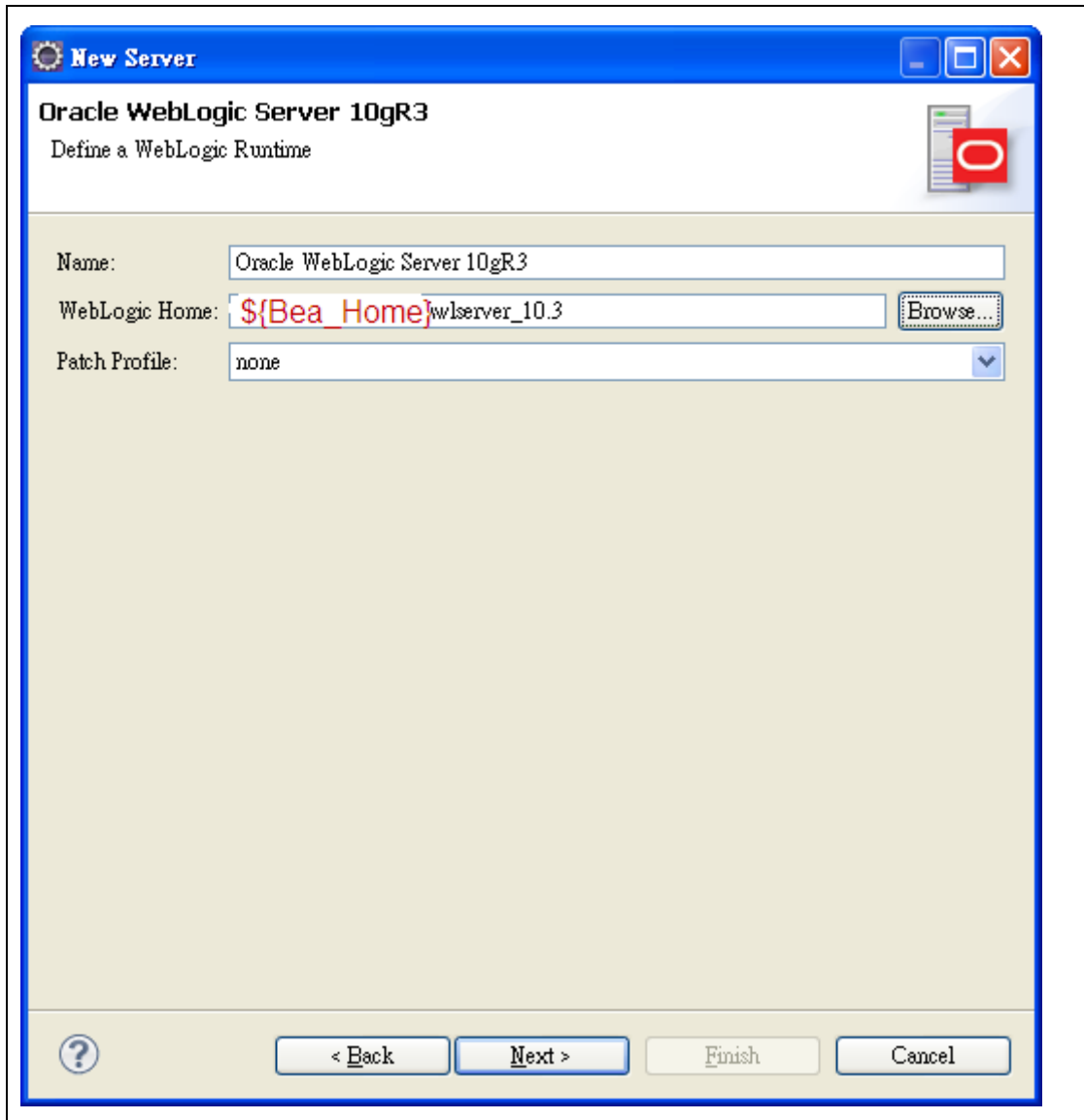
實作

閒話休提，讓我們改快著手開發為要，本公司代理的是世界第一品牌的 Java EE application Server，WebLogic，就讓我們使用 WebLogic 10.3 這個版本來作為我們開發程式的部署平台吧；至於開發工具，我們也從眾選擇 Eclipse 3.5；另外為了順利開發 EJB 我們需得由 OTN(http://www.oracle.com/technology/software/products/oepe/oepe_11g.html)下載 oepe-ganymede-11.1.1.1.200904131333.zip，將開檔打開，將其中 eclipse folder 下的 plugins 與 features 兩個 folder 拋到你自已個 Eclipse 3.5 目錄，重啓 Eclipse 即可。

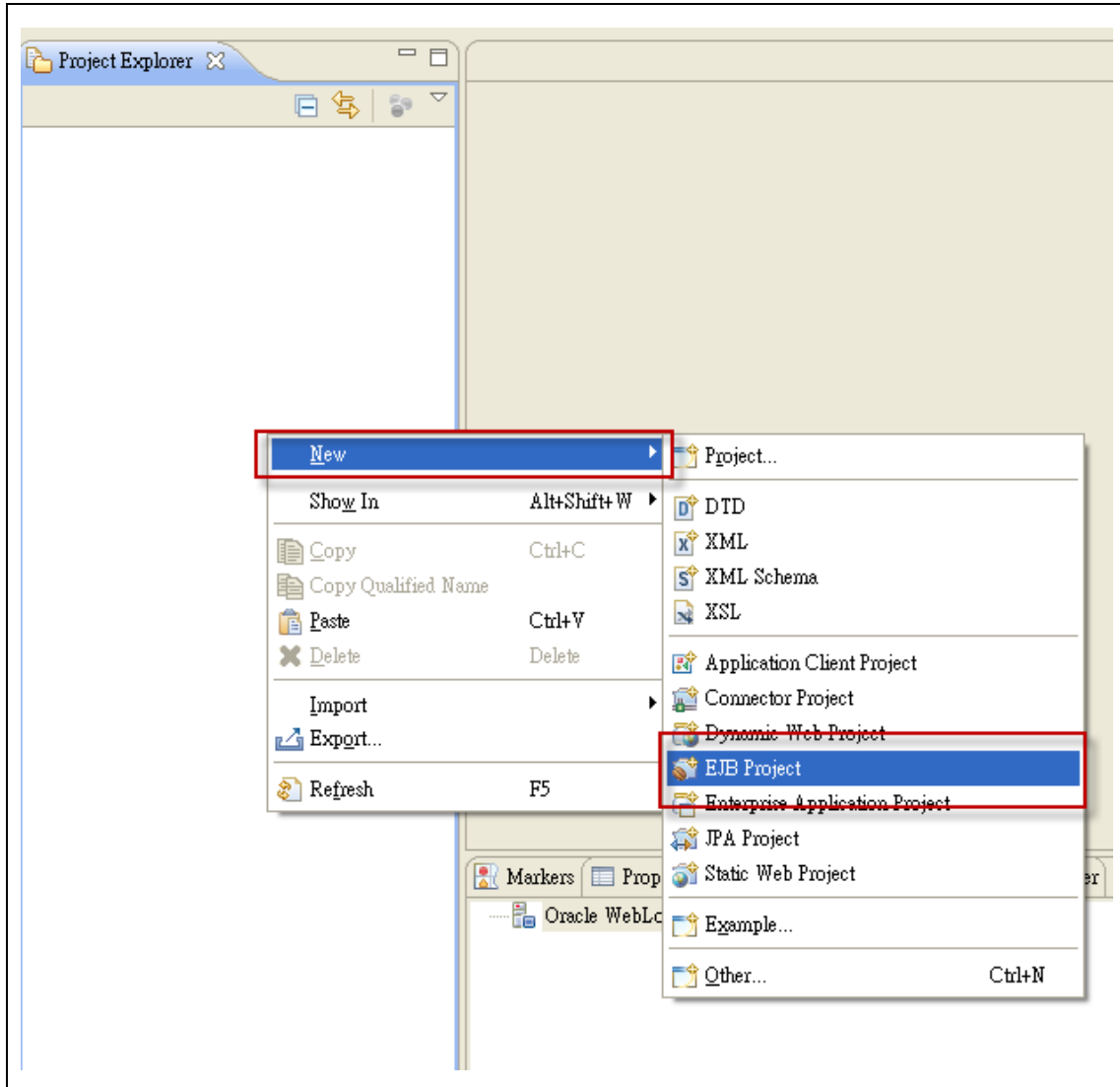
建置一個 Server Config

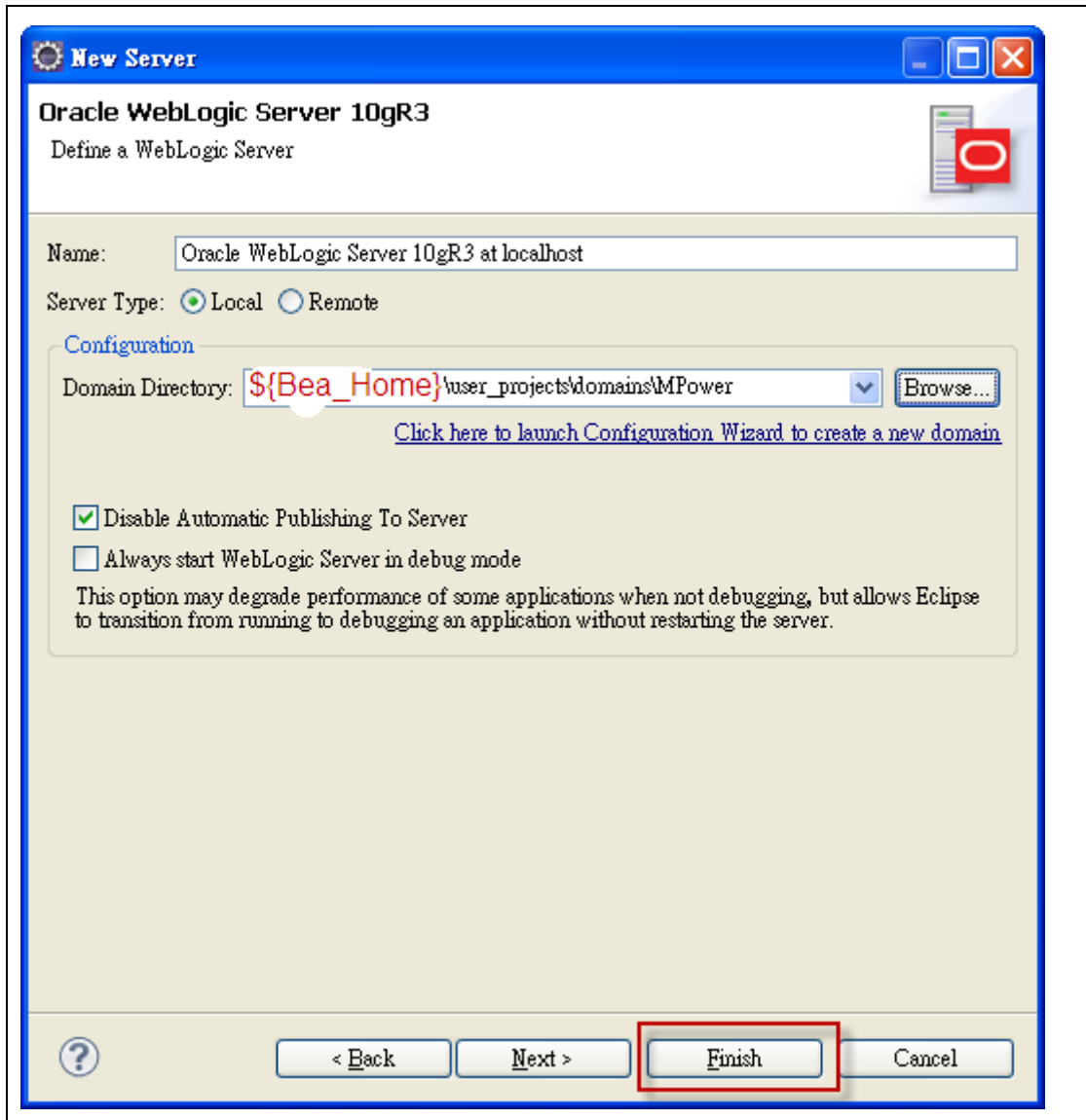
為了讓 EJB 的專案可以在建置時 include weblogic 本身的相關 lib，請在 Eclipse 畫面下方的 server 頁籤點按右鍵選擇 New/Server，先建置 server config 如下：

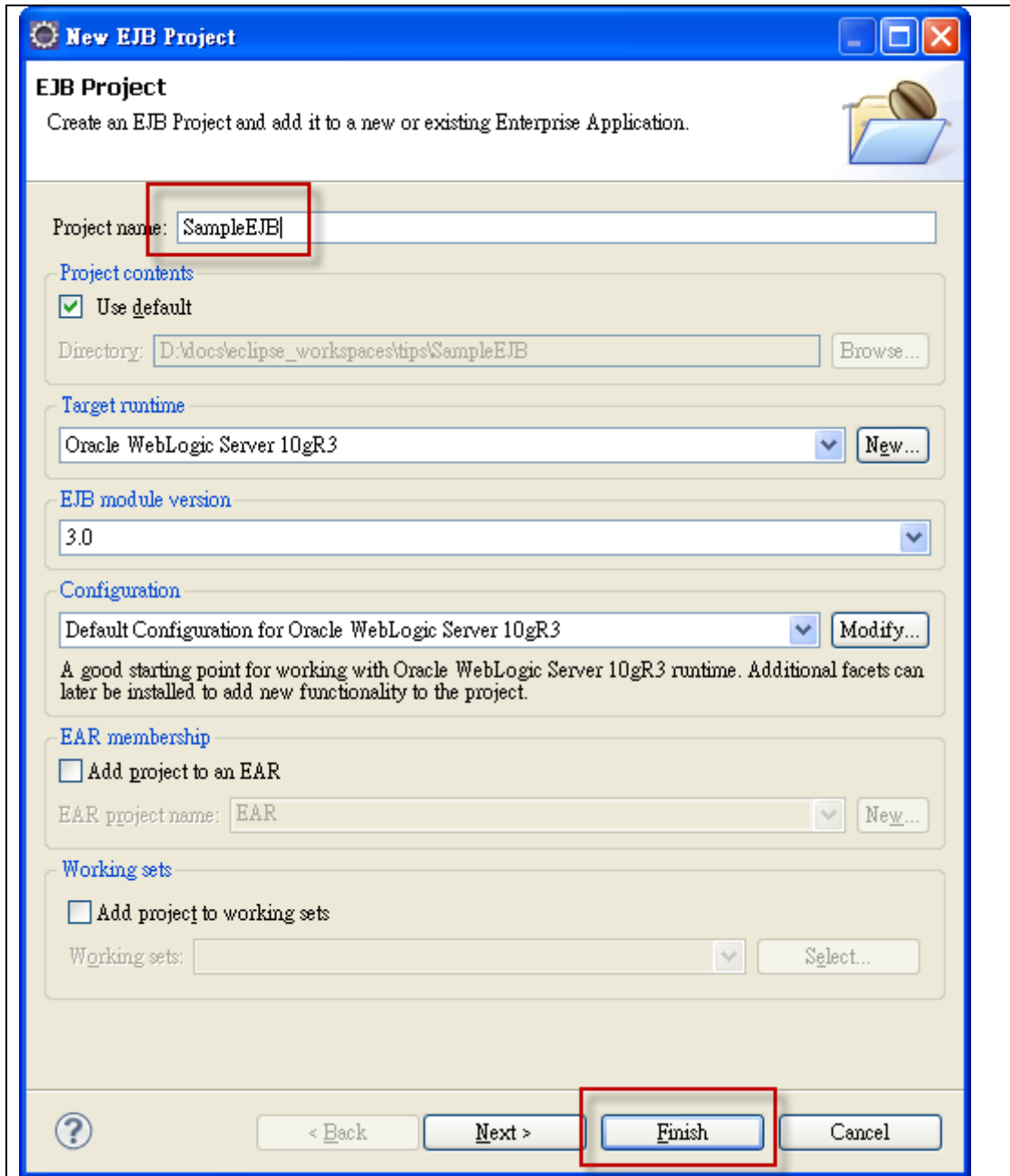




撰寫 EJB



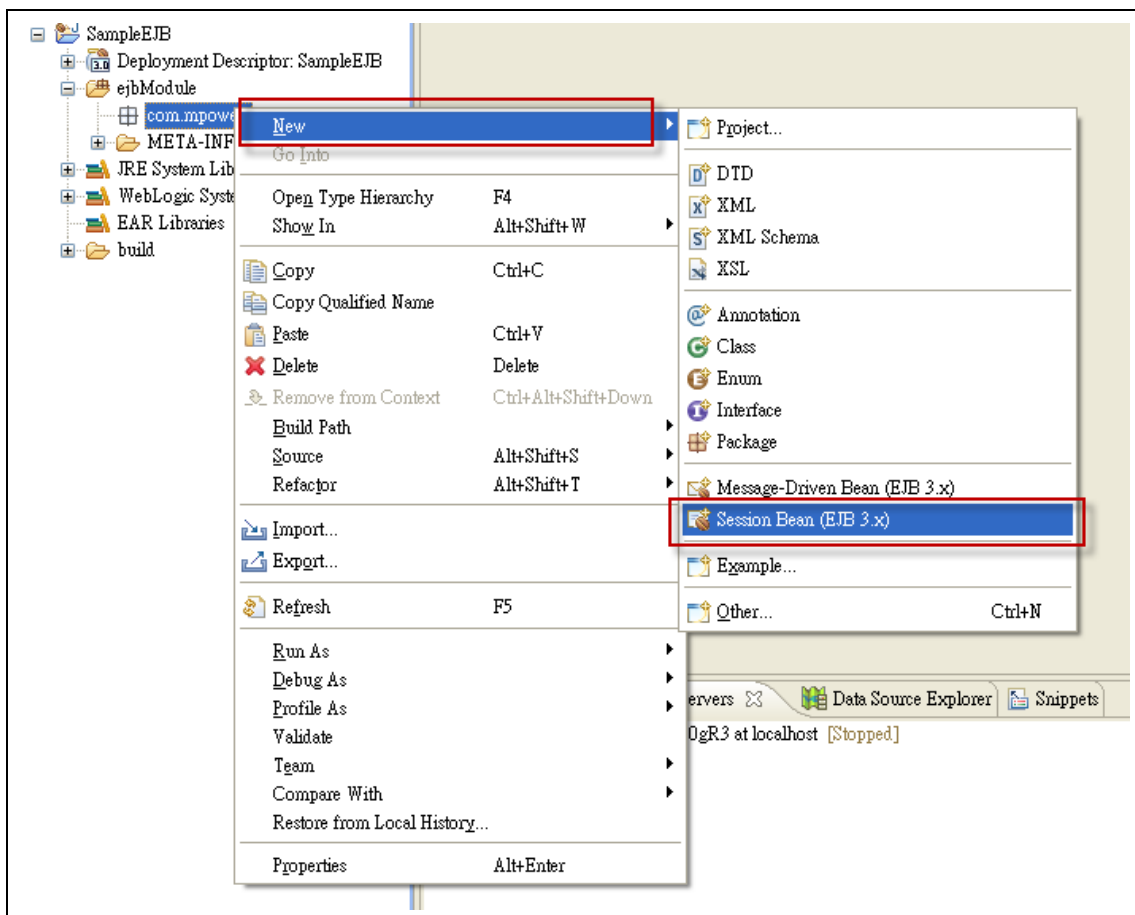




之後請先建個 package



之後在該 package 上點按右鍵，選 New/Session Bean(EJB 3.x)



依如下圖內容填寫

Create EJB 3.x Session Bean
Specify class file destination.

EJB project: SampleEJB

Source folder: /SampleEJB/ejbModule

Java package: com.mpower

Class name: Echo

Superclass:

State type: Stateless

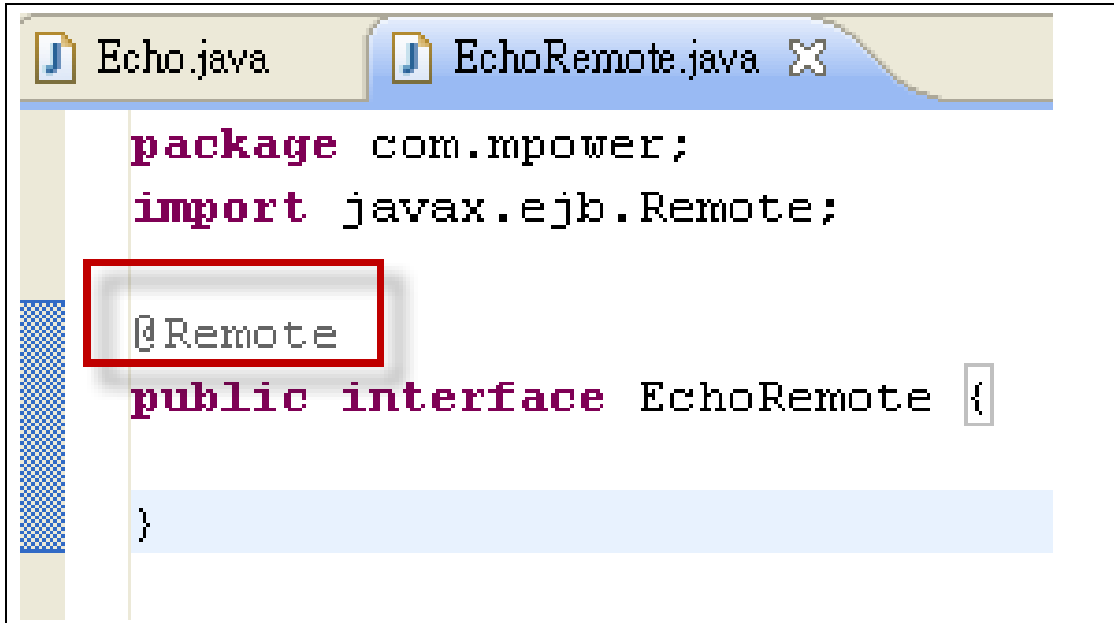
Create business interface

Remote com.mpower.EchoRemote

Local com.mpower.EchoLocal

之後會產生出兩個檔案，一個是真正撰寫邏輯的 Echo，另外一個是 Echo 的 Interface，這個 Interface 可不是什麼 Home Interface 或是 Remote Interface，他只是一個簡單又一般的 Java Interface，並沒有特別繼承其他的 Interface，這點很重要，他代表的是 EJB3.0 像 Spring Framework 的概念看齊。

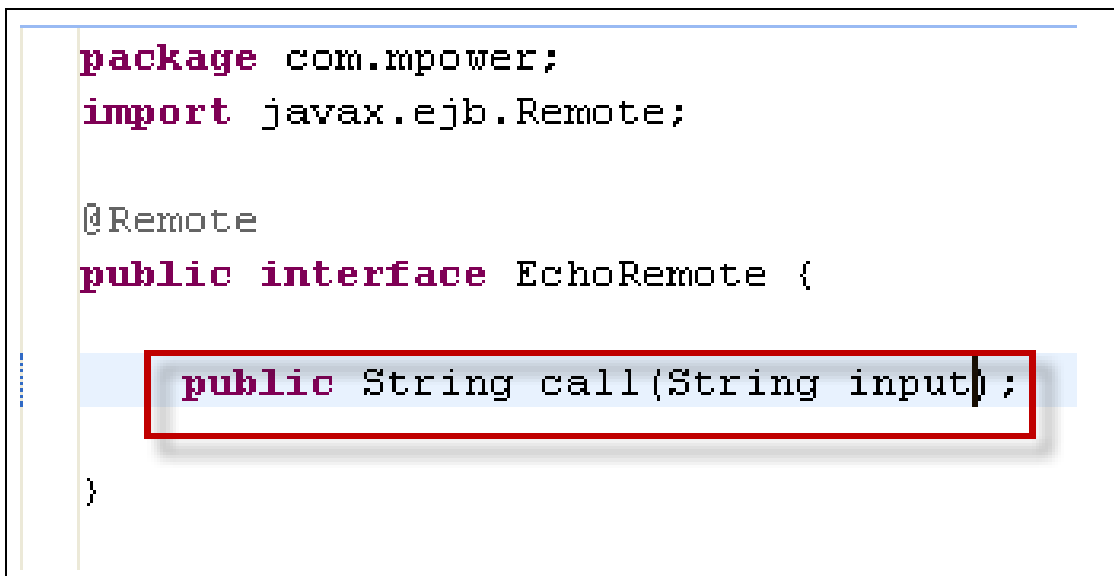
我們再來看看 EchoRemote 介面的內容。



```
package com.mpower;
import javax.ejb.Remote;
@Remote
public interface EchoRemote {
}
```

紅色的框框代表的就是我們主題所說的 Annotation，這個特別的標籤描述了一個 class 如果 implement 這個 Interface，那個他就一個可以透過網路來進行呼叫的 EJB(反之如果僅可供 Server 內的元件呼叫的 EJB 使用的就是 @Local Annotation)。

讓我們替這個 Interface 加個 method 吧



```
package com.mpower;
import javax.ejb.Remote;

@Remote
public interface EchoRemote {
    public String call(String input);
}
```

在我們加了 method 後，另外一支程式 Echo，由於有 implement EchoRemote，所以會被要求也一併 implement 這個 EchoRemote 新增加的 method call，就讓我們補上吧。

```
package com.mpower;

import javax.ejb.Stateless;

/**
 * Session Bean implementation class Echo
 */
@Stateless
public class Echo implements EchoRemote {

    /**
     * Default constructor.
     */
    public Echo() {
        // TODO Auto-generated constructor stub
    }

    @Override
    public String call(String input) {
        return "echo for : " + input;
    }
}
```

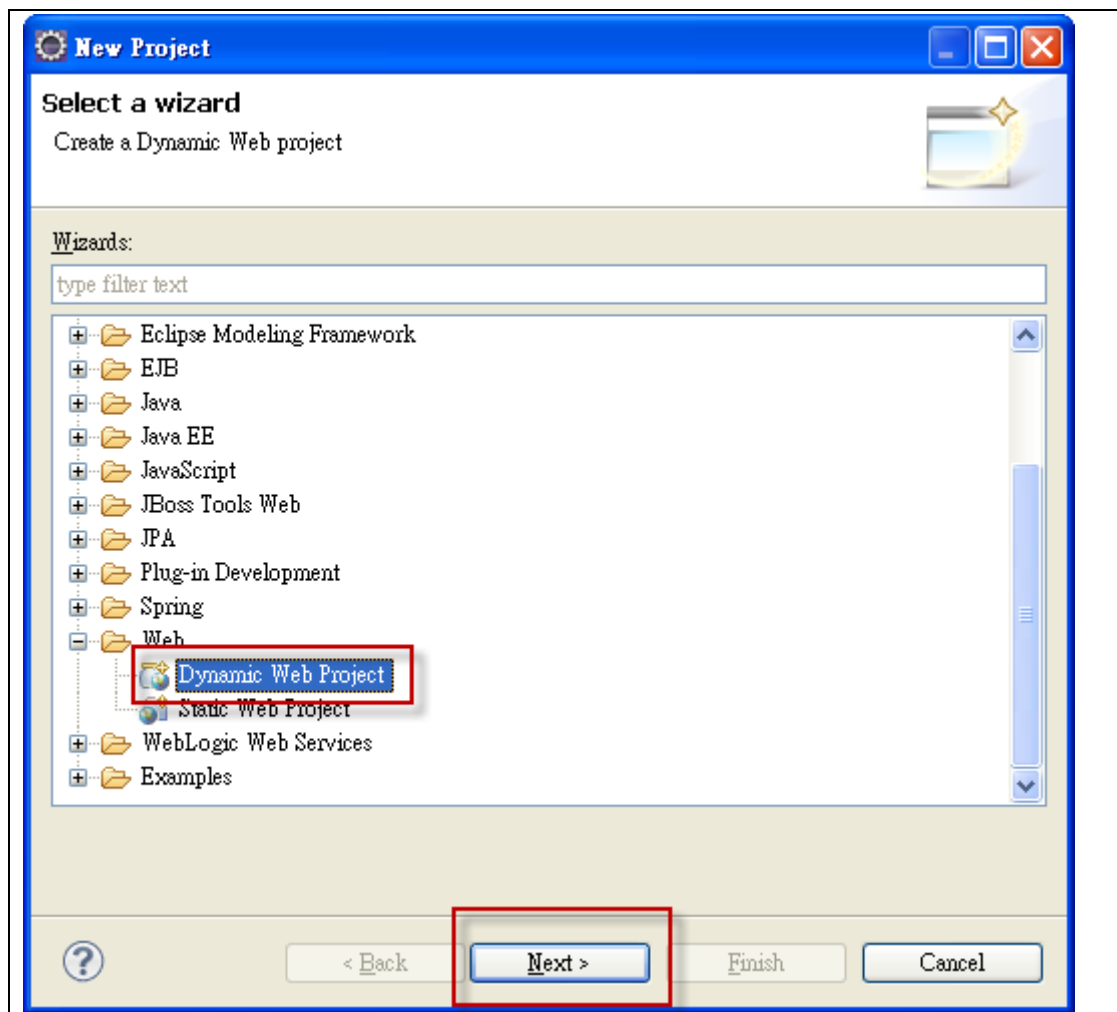
簡單補上了 call 的實作後，我們順便看到 Echo 本身也被宣告了 @Stateless 這個 Annotation，這是標註這個 EJB 是屬於 Stateless Session Bean。

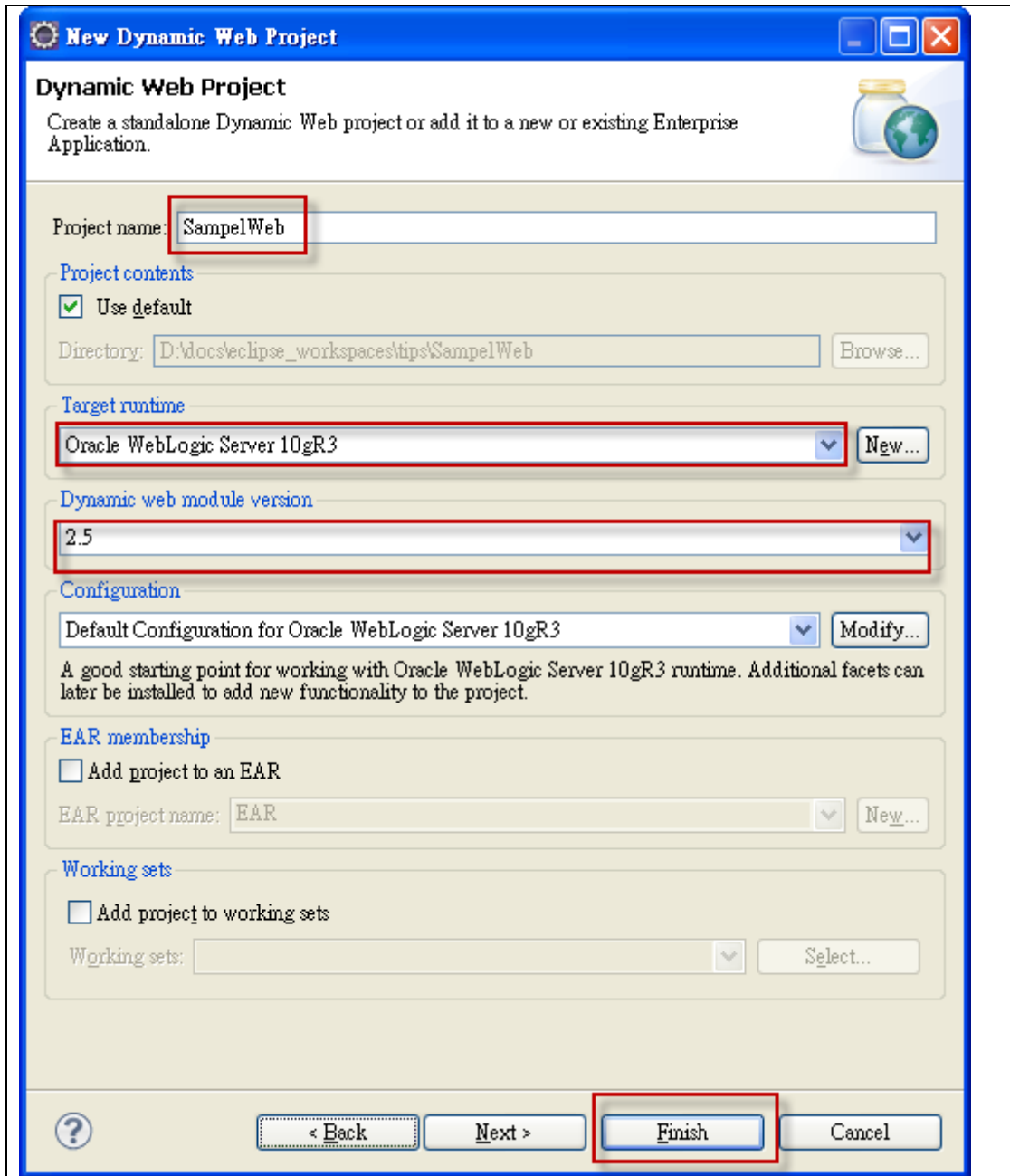
不用懷疑，一個 Stateless Session Bean 已經完成，Eclipse 所做的行為只是替我們的 EJB 加上了一個為了設計架構的彈性所製造出來的 Interface，Echo 只是個 POJO，而 EchoRemote 也只是一個沒有繼承其他 Interface 的簡單 Interface，全部的全部，除了 Annotation 之外，與我們平時寫的 POJO 程式沒有不同，這就是 EJB3.0 所要表達的精神。

寫完測試

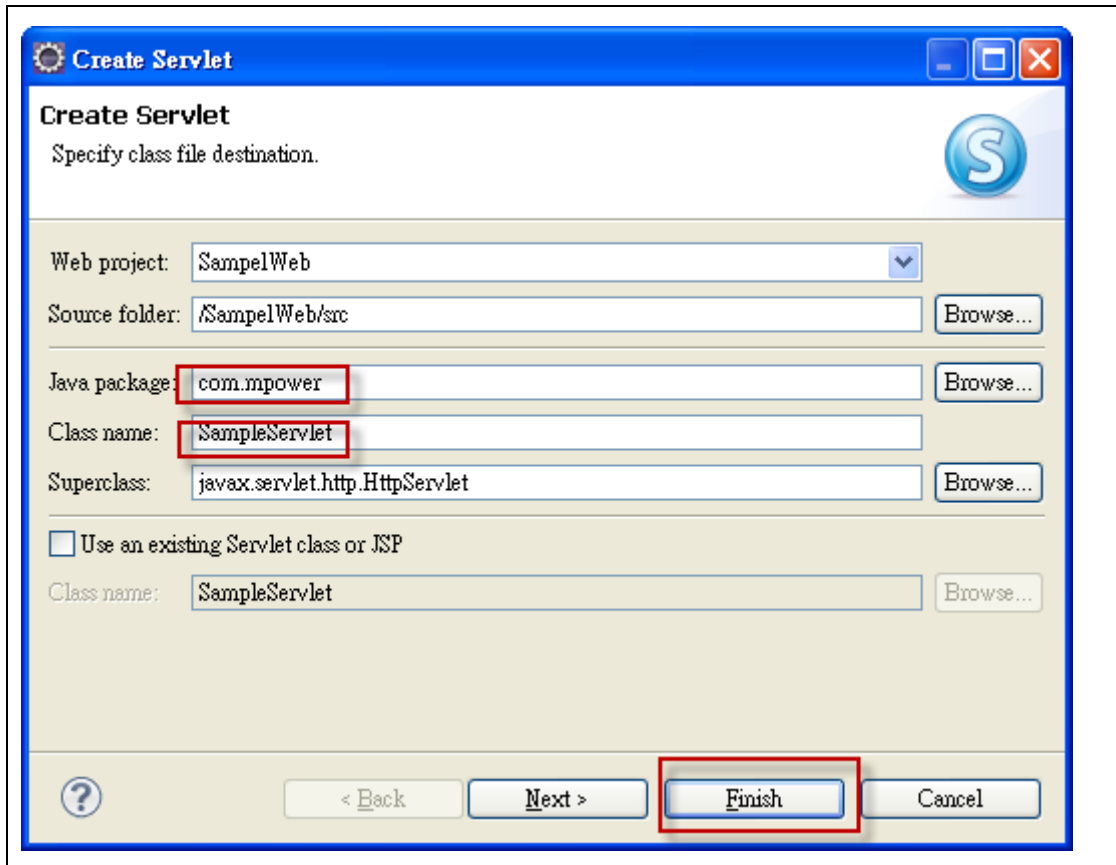
寫完當然要測，沒測沒真相，Annotation 再度發威；如果我們想要在 servlet 中呼叫一個 EJB，在 EJB3.0 前的年代，我必須要先使用 JNDI 將 EJB Home Interface 取得，再用它來 create 一個 Remote Interface，再用這個 Remote Interface 來當作 EJB 的代理，呼叫 Remote Interface 上的 business method...，現在，全部不用，以我們這個範例來說，在 servlet 中宣告一個 instance variable，型態就是 EchoRemote，並且在上面加註個 Annotation，servlet 的變數會被自動 inject 進 EJB 的 instance。

請新增一個 Dynamic Web Project 如下





之後會產生 SampleWeb Project 在 Eclipse 上，請在該 Project 點按右鍵，選取 properties 後，進行如下設定將既有的 SampleEJB 加入到 SampleWeb 的 classpath 中，以便可以引用該 Project 中的 EchoRemote Interface。



在該 servlet 上宣告一個 EchoRemote 型態的變數，在其上加個 @EJB 的 Annotation，並簡單撰寫邏輯，邏輯中當然直接引用 EchoRemote 型態的變數即可；注意，我可沒有做任何的 JNDI call，搞那些 Home 與 Remote Interface 的玩意兒，我的 EchoRemote Interface 可是簡簡單單，沒有 extend 其他怪異 Interface 的 Interface。

```
package com.mpower;

import java.io.IOException;

/**
 * Servlet implementation class SampleServlet
 */
public class SampleServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

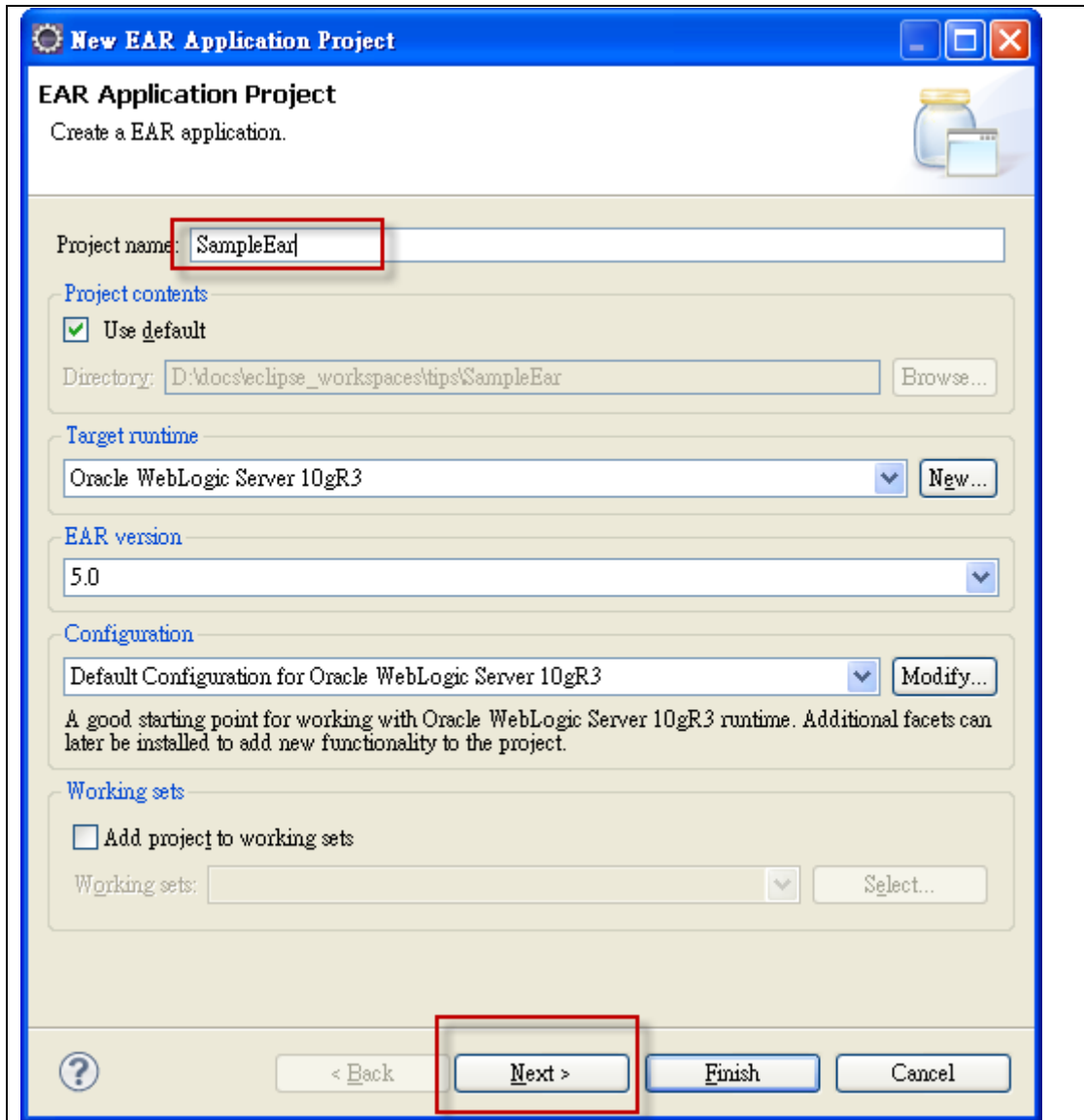
    @EJB
    private EchoRemote echo;

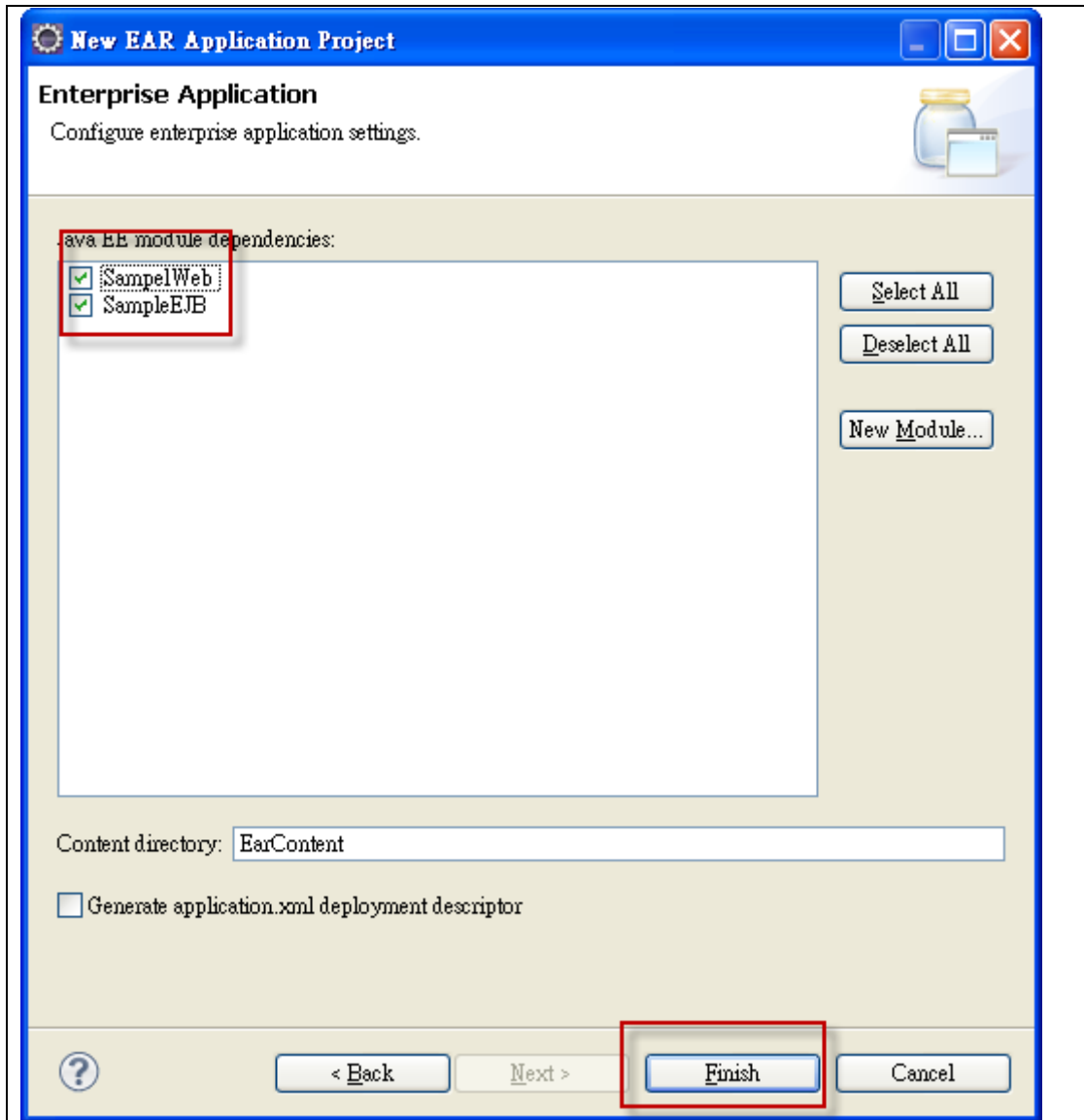
    /**
     * @see HttpServlet#HttpServlet()
     */
    public SampleServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

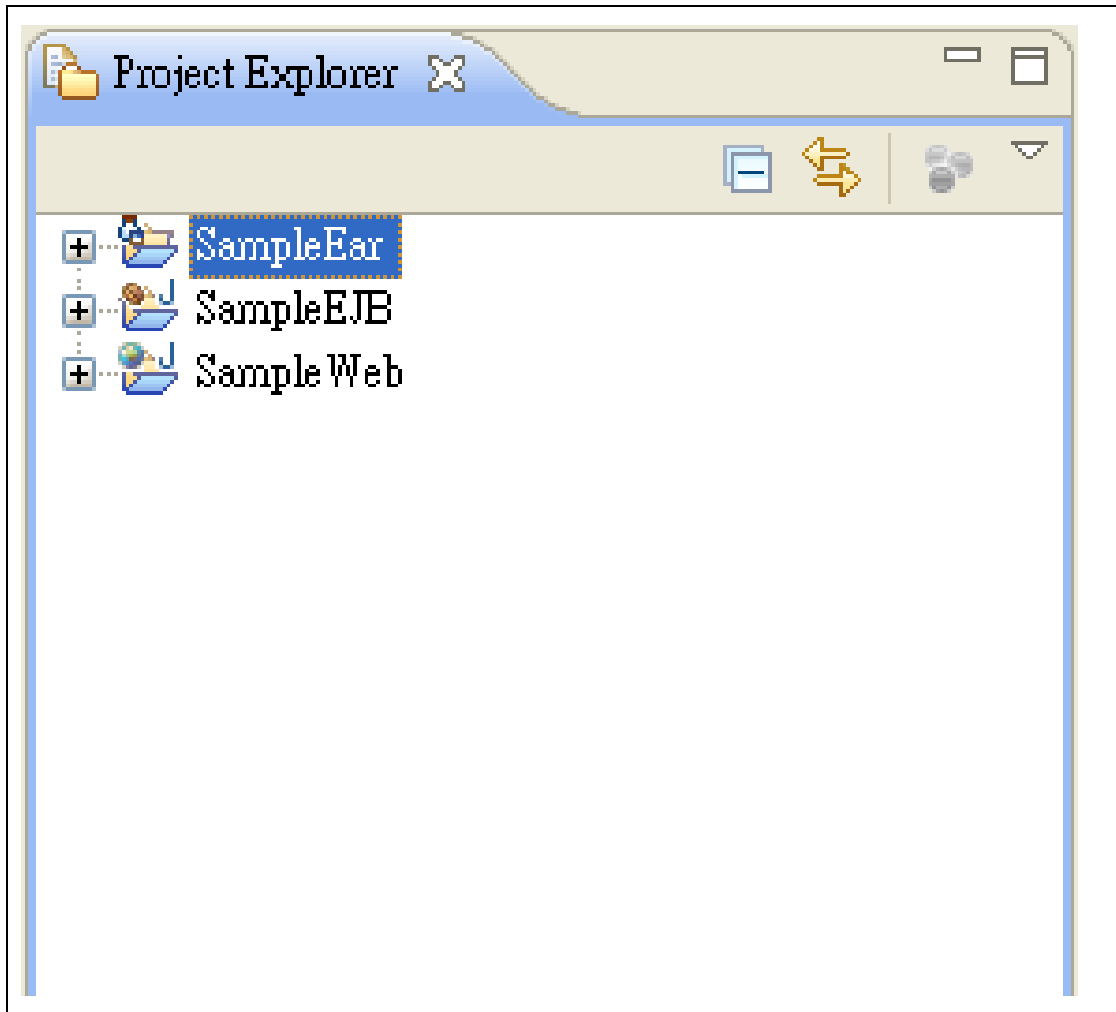
    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     * response)
     */
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        String rt = echo.call("hello");
        response.getWriter().println(rt);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
     * response)
     */
}
```

當然我們必須建置一個 Enterprise Application Project 將這個 Web Project 與 EJB Project include 進來。







將 SampleEar 匯出為一個 Ear 檔，再來，請啓動 WebLogic Server，使用 Admin Console 將這個 Ear 檔部署在 server 上，並在瀏覽器上打入對應這個 servlet 的 url，我想有圖有真相，各位當知我所言不虛阿。

