Java+EJB+XML+資料庫

整合應用程式設計參考題解

莊幸隆

本書參考題解，含部份習題題解之重點提示或章節參考，專題設計之類的問題，學員生可分組當學期的程式編寫專案。

準備事項

為了能另外新建檔案夾路徑來存放與使用自行製作的 Servlet 及 JSP 程式，在網站伺服器的 Tomcat-4.0.4 以上版本中，於%TOMCAT_HOME%\conf\server.xml 內(在 Unix 環境則為$TOMCAT_HOME/conf/server.xml)，可加進下列有灰色背景的一段：

```
…
<!-- Tomcat Manager Context -->
…
```

```
<!-- Tomcat local Context -->
<Context path="/local" docBase="local" debug="0"
                       reloadable="true" crossContext="true">
    <Logger className="org.apache.catalina.logger.FileLogger"
            prefix="localhost_local_log." suffix=".txt"
            timestamp="true" />
</Context>
```

```
<!-- Tomcat Examples Context -->
…
```

之後在%TOMCAT_HOME%\webapps 檔案路徑內，即可新建 local 檔案夾目錄，再隨己意應用在 local 檔案夾目錄中，增加其他子目錄或檔案。

J2EE Srver 則應熟悉使用建置工具，用於製作 EJB 以及 Web 元件的 Servlet 及 JSP 之類程式，例如像 deploytool 或 packager 等工具。並可在 J2EE Server 的 web.properties 去修改成 documentroot=/或者其他的設定，以配合規劃欲建立的應用程式伺服作業平臺。

Tomcat 及 J2EE Server 下載之網址各為

www.apache.org
java.sun.com

第一章

〔練習題〕試說明 Java 為何稱是“write once, run anywhere”。

Write once, run anywhere 之意義，參考圖 1.2，圖 1.3，並從 Java 程式執行的要件是甚麼去思考。如果說當今所有電腦作業平台僅只有一種，程式語言及編譯器也只有一種，則大概不用強調，那必然是 Write once, run anywhere，而現行之各式多樣的作業平台便需”通用”的執行環境，方能夠一次開發隨處可用，Java 執行環境 JVM (Java Virtual Machine)的製作，即是以”通用” 作業平台的觀念設計，支援“write once, run anywhere”的應用。

〔練習題〕試將 Java application 的應用程式中 `main` 程式如改為

```
static public void main(String[] args) {
…
```

會有何不同，並說明之。

`public` 表示是方法程式的運用別，`static` 表示是方法程式不需先建立工作體物件，便能直接由 JVM 啟動叫用，本練習題

```
static public void main(String[] args) {
…
```

與一般範例的

```
public static void main(String[] args) {
…
```

兩關鍵字使用上，執行結果相同，但實際應用順序上，應參考 2.4 節的詳細說明。

〔練習題〕試行觀察 *Aloha.jsp* 在瀏覽器的使用，於首次載入與後續重新載入，有何明顯差異。再比較與 *AlohaServlet.java* 程式載入執行又有什麼不同。

Aloha.jsp 網頁在瀏覽器的使用，於首次載入與後續重新載入，最明顯的差異便是回應的速度，因為首次載入，JSP 網頁需先被編譯成 Servlet，然後常駐於主記憶體中，後續的使用，會直接由主記憶體中引用，同樣地，*AlohaServlet* 也會直接由主記憶體中引用，這正是別於一般 CGI 方式的每次需重載重譯。JSP 或 Servlet 程式對資源的運用率可較優。

〔論述題〕試比較說明 Java application, Applet, Servlet, 以及 JSP 的差異。

可從表象上和應用上的角度來區分 Java application, Applet, Servlet, 與 JSP 的差異，請參考 1.4 節與 1.5 節各類範例，以及作業平台的需求。

〔練習題〕中國農民曆年數計法以 10 天干 (甲乙丙丁戊己庚辛壬癸) 12 地支 (子丑寅卯辰巳午未申酉戌亥)，排序為甲子年、乙丑年、丙寅年、…、壬戌年、癸亥年，成六十年一輪，西元 2000 年為農民曆的庚辰年，試以一轉換程式，鍵入西元年，程式則回覆農民曆年。

東方農曆與西曆的計算法，一以月亮為曆(陰曆，lunar year)，一則以太陽為曆(陽曆，solar year)。由於量測會有誤差情形，月曆以潤月，日曆以潤年修正誤差。月曆年與日曆年日期雖然不是完全同等計算，但可以約略的換算。月曆以六十年一甲子周而復始，西元第一年是辛酉年，第二年是壬戌年，西元前的月曆算法依樣類推。本

題作法可以將下式組合成六十個不同的年，存為陣列方式，再利用其年份除以六十後的餘數，作為註標對應六十個年，取得月曆年份名列示。

```
String[] s = {"甲乙丙丁戊己庚辛壬癸",
              "子丑寅卯辰巳午未申酉戌亥"};
```

〔練習題〕同樣地試寫一程式用來計算地球表面積。(提示：先找出球體表面積計算的公式，而地球的表面積約略是為 510,100,934 平方公里)

球體表面積計算公式：

$$A = 4pr^2$$

然地球科學及數學的應用，計算地球表面積，其結果可能有數種答案，本題的目的僅在做程式練習，精確的算法可與台北市立天文科學教育館請教。

〔練習題〕光年為長度量距單位，即光在真空中進行一年所走的距離為一光年，那麼準備一個程式，計算一光年可繞地球幾圈。(提示：先找出光速值)

光速的數值：$c = 3*10^8$ m/s，即約每秒 30 萬公里(299792.458 公里)。

一光年約為 9.46053×10^{12} 公里，可套用於計算繞地球幾圈。

第二章

〔論述題〕short 及 char 資料型式如何互換，並舉例說明。

short 和 char 類型同為 16-bit，並注意 short 可表達的最大數值。

〔論述題〕如何將一個 char 字元資料型式轉為 String 字串類別型式，例如字元 'A' 轉為字串"A"。

char 是基本資料類型，String 是類別類型。簡單方式是字元與字串相加後的運算式，會產生字串的結果。

〔論述題〕試說明 throw 和 throws 的差異，並舉例說明。

throw 用於敘述式，throws 用於方法程式。範例參考 3.7 節。

〔論述題〕試說明 this 和 super 的差異，並舉例說明。

this 和 super 的差異，參考 2.4 節。

〔專題設計〕依前面章節所敘述，數目值有其最大及最小的限制，想像如何去製作一個數目值無最大及最小限制的類別型式程式，以便給需要數目值不限制的程式來使用。[Griswold(1997)]

本專案題旨在設計一個數目值無最大及最小限制的類別型式程式，可參考[Griswold(1997)]。本題的目的在啟發另類的思考，打個比方，當數目值超過基本資料型式的 long 能表達的最大值時，是否應出現設計 double long 的資料型式(128-bit)？或甚至出現 quad long 的型式(256-bit)？還是說乾脆設計個數目值沒限制的資料型式或類別？並同時藉以此思考法，找出更多的專題或論文題目。

〔練習題〕說明不同的資料型式或類別型式作指派設定值 assignment 時，可以容許的方式。

不同的資料類型或類別類型作指派設定值 assignment 時，可以容許的方式，參考 2.14 節。

〔練習題〕製作 Java 使用手冊文件是開發程式很重要的一環，試先行參考附錄 A 的範例，作些簡單的 Java 文件。

本練習題可配合閱讀第六章及附錄 A 相關部分，先進行了解。

第三章

〔論述題〕試說明 while 敘述式和 do-while 敘述式的差別。

while 敘述式和 do-while 敘述式的差別參考 3.4 節。

〔練習題〕試寫一兩數相乘，可以偵測出 OverFlow 及 UnderFlow 的方法程式。

```
  static int multiply(int multiplicand, int multiplicator)
      throws OverFlowException, UnderFlowException {
  …
  }
```

兩數相乘，偵測 OverFlow 及 UnderFlow 的方式，參考 2.15 節與 3.7 節。

〔練習題〕在 BankAccount 的例子，當碰到提款金額大於存款餘額時，應如何修改程式，丟出例外狀況。

提款金額大於存款餘額時，丟出例外狀況參考 3.5 節與 5.5 節。

第四章

〔**論述題**〕**試作** char **以及** String **宣告的同時也設定初值。**

作 char **以及** String **宣告的同時也設定初值的範例：**

```
String[] s = {"甲乙丙丁戊己庚辛壬癸",
              "子丑寅卯辰巳午未申酉戌亥"};

char[] c = {'登','泰','山','而','小','天','下'};
```

如有多維陣列初值的設定，參考 4.2 **節。**

〔**論述題**〕**試說明陣列可包含那些資料型式。**

陣列可包含那些資料型式，參考 4.1 **節。**

〔**論述題**〕**陣列宣告的語法除前述外，另有一種方式為：**

```
int []a;
```

試行說明有何差異。

整理一下陣列宣告的方式，便可瞭解其相同之處，例如：

```
int[] i;
int i[];
int []i;
```

以及其它不同的資料與類別型式。重點在語法的設計，能接受這些不同的方式。

〔論述題〕試說明陣列拷備複製與科融複製的不同。

陣列拷備複製與科融複製的不同，參考 4.1.2 小節。

〔練習題〕如果有機會用到四維的初始值設定，試用推論及想像力，那麼四維浮點小數陣列應如何設定初始值？

```
    double[][][][] Q = ……;
```

四維浮點小數陣列初始值的設定，需注意外括深度的組數：

```
double[][][][] Q =
    {   {   {   {0.1, 1.0}
            },
            {   {0.2, 2.0}
            }
        },
        {   {   {0.3, 3.0}
            },
            {   {0.4, 4.0}
            }
        },
        {   {   {0.5, 5.0}
            },
            {   {0.6, 6.0}
```

```
            }
        }
    };
```

〔練習題〕試作另一類的陣列相乘，讓陣列 C 由下式求得：(提示：首先應知道陣列 C 怎麼求得)

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \end{pmatrix}$$

陣列相乘，首先要找出各元素值的求法，然後參考 4.2.1 小節。

〔思考題〕先了解 class 類別資料型式與其他程式語言，例如 C 或 Pascal 的 struct, union 或 record 的觀念、形式之異同，說明 Java 為何沒有沿用。

在 C 語言的 struct, union 或 Pascal 語言的 record，其資料類型也是一種自訂的複合型式，在 Java 或 OO 語言，用類別來定義宣告複合型式的資料類型，是一種自然的作法，因類別本身就是結構式的複合資料，故基於程式語言的簡單化，直接以類別方式引用，最為適切了。

第五章

〔論述題〕試說明 overloading 和 overriding。

overloading 參考 5.7 節，而 overriding 則是子類別對於母類別原有的方法程式，在子類別內進行翻寫，並應用於子類別。

〔練習題〕試以遞歸式方法程式的解法，來求解兩個數目的最大公約數 GCD(Greatest Common Divisor, 或 GCF, Greatest Common Factor)及最小公倍數 LCM(Least Common Multiplier)。

先整理出最大公約數 GCD(Greatest Common Divisor, 或 GCF, Greatest Common Factor)及最小公倍數的表達方程式。

〔練習題〕試自行找出可以遞歸式解法的題目，並逕行撰寫其程式。(這是一個自行研習最好的練習)

自行找出可以遞歸式解法的題目，首重找出能以用方程式表達的，便會容易套用遞歸方法。

〔思考題〕河內之塔在搬移的次數上，金環總數在一個金環時，僅須移動一次，兩個金環時，須移動三次，三個金環時，須移動七次，如此下去，那麼照 64 個金環時，須移動的次數會多少？如果假設每移動一次需時一秒，那麼 64 個通通由起點柱全部移到終點柱要多久？假設用電腦模擬移動，每一個移動需時百萬分之一秒的話，那需時多久？(提示：先將公式整理出來)

整理出河內之塔金環搬動計次的公式，便可以求得耗時的多寡。下示

$$2^n - 1 \qquad \text{where } n > 0$$

可藉以想像 $2^{64} - 1$ 秒會多久。另以電腦模擬移動的百萬分之一秒每動需時會多久。

第六章

〔論述題〕何謂 class？何謂 object？試詳述重要觀念與關係。

Class 是一種類別的型式，簡單化思考，可視之為基本資料型式的同等地位，然其複合資料型式的運作，功能較強也較複雜。Object 則是 `class` 類別的實際工作體，可簡單化地視為基本資料型式宣告的變數之同等地位。工作體的運算藉其方法程式進行，就像基本資料型式的變數藉運算符號來演算。

〔練習題〕試再找出五個其他類抽象觀念的 abstraction。(提示：太多了，除前述外，譬如地震的芮氏震級，米制長度量度，燈泡亮度瓦特，…)

抽象觀念的 abstraction，藉量度單位方式和非量度方式表示，甚至美或帥，好吃不好吃，快跟慢，深及淺，濃淡，難度，都可能做量度或非量度方式的抽象表達。

〔論述題〕何謂 `abstract` 抽象類別？

`abstract` 抽象類別，參考 6.3.2 小節。

〔論述題〕試說明 `abstract` 類別型式與 `interface` 介面的差異。

`abstract` 及 `inetrface` 的差異，由宣告，應用等的差異，參考 6.3.4 小節。

〔論述題〕試說明 `Vector` 與 `ArrayList` 類別的差異。

參考 6.3.7 小節說明 `Vector` 與 `ArrayList` 類別的差異。並應留意安全的線程作業 Thread-safe 與非 Thread-safe 和 synchronization 同步的關係。本題儘往功能和應用形式方向去思考。

〔論述題〕試說明使用 Singleton 的主要目的。

參考第八章，第十四章相關部份。

第七章

〔論述題〕方法程式 write 與 print 兩者之間有何異同？

write 與 print 兩者之間的異同，參考 7.3.1 小節。

〔思考題〕如果一個程式完全沒有資料輸出入的封閉式運作，依推測這會是什麼樣的程式？

一個程式完全沒有資料輸出入的封閉式運作，那可能什麼都是，也可能什麼都不是的程式。如果是有意義的這類程式，那也大概無法知曉是何等挑戰的運算作業了？

〔練習題〕試利用標準輸出入，由鍵盤輸入文字元，螢幕顯示 Unicode 的 16 進位制碼。

由鍵盤輸入文字元，螢幕顯示 Unicode 的 16 進位制碼，此為 short 和 char 類型互換的練習，參考 7.5 節。

〔論述題〕byte stream 轉 character stream 是如何運作的？

byte stream 轉 character stream 的運作方式，參考 7.3.1 小節，重點是如何利用類別設定轉換橋樑，完成資料串流類別型式的變換。

〔練習題〕試說明下示使用方式與前兩例的差異(7.4 節)：

```
PipedReader pr = new PipedReader();
PipedWriter pw = new PipedWriter(pr);

PipedInputStream pis = new PipedInputStream();
PipedOutputStream pos = new PipedOutputStream(pis);
```

雖然產生的結果會相同，一是"寫入"類的物件工作體當"讀取"類的參數，另一則是"讀取"類的物件工作體當"寫入"類的參數，重點是配合的"寫入"類與"讀取"類兩者之間建立的 Pipe **供此兩類的物件工作體使用。**

```
PipedWriter pw = new PipedWriter();
PipedReader pr = new PipedReader(pw);

PipedReader pr = new PipedReader();
PipedWriter pw = new PipedWriter(pr);
```

character mode

```
PipedOutputStream pos = new PipedOutputStream();
PipedInputStream pis = new PipedInputStream(pos);

PipedInputStream pis = new PipedInputStream();
PipedOutputStream pos = new PipedOutputStream(pis);
```

byte mode

〔**論述題**〕**如何將** `String` **轉為** Stream，**並舉例說明。**

將 `String` **轉為** Stream，**不論是轉作** character **或是** byte stream，**運用本章所談論過各類類別相關讀寫的方法程式，即可轉換，本練習題主要在對這兩個名稱，能有**

較清楚的認知與了解。

第八章

〔**論述題**〕**試說明** wait **和** sleep **方法程式有何不同？**

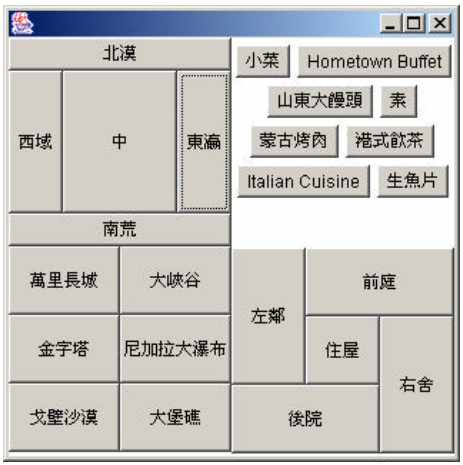
wait **和** sleep **方法程式有何不同，參考** 8.3 **節**。

〔**論述題**〕**試說明** ThreadLocal **和** InheritableThreadLocal **類別兩者的區別**。

ThreadLocal **和** InheritableThreadLocal **類別兩者的區別，參考** 8.4.2 **小節與** 8.4.3 **小節**。

第九章

〔練習題〕試設計一個視窗版面包含前述幾個版面如下示視窗畫面：(9.5 節)



將原來每一個版面當一個組件看待進行，便能取用適當的編排方式，像採格列式：

```
import java.awt.*;

public class ComboExample extends Frame {
    public static void main(String[] args) {
        ComboExample ce = new ComboExample();
        ce.add(new ComboPanel());
        ce.addWindowListener(new WinCloser());
        ce.setSize(360,360);
        ce.setVisible(true);
    }
}

class ComboPanel extends Panel {
```

```
    public ComboPanel() {
        setLayout(new GridLayout(2, 2));
        add(new BorderLayoutPanel());
        add(new FlowLayoutPanel());
        add(new GridLayoutPanel());
        add(new GridBagLayoutPanel());
    }
}
```

〔練習題〕試設計一個視窗版面編排，按鈕安置如下視窗畫面：

我思故我在版面上的格子位置，找出適用的編排方式：

```
import java.awt.*;

public class IthinkExample extends Frame {
    public static void main(String[] args) {
        IthinkExample ie = new IthinkExample();
        ie.add(new IthinkPanel());
        ie.addWindowListener(new WinCloser());
```

```
        ie.setSize(180,180);
        ie.setVisible(true);
    }
}
class IthinkPanel extends Panel {
    public IthinkPanel() {
        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();

        // 應用到所有元件的條件
        gbc.fill = GridBagConstraints.BOTH;
        gbc.weightx = 1.0;
        gbc.weighty = 1.0;

        // 應用到個別元件的條件
        addButton(new Button("我"), gbc, 0, 0);
        addButton(new Button("我"), gbc, 4, 0);
        addButton(new Button("思"), gbc, 1, 1);
        addButton(new Button("思"), gbc, 3, 1);
        addButton(new Button("故"), gbc, 2, 2);
        addButton(new Button("我"), gbc, 1, 3);
        addButton(new Button("我"), gbc, 3, 3);
        addButton(new Button("在"), gbc, 0, 4);
        addButton(new Button("在"), gbc, 4, 4);
    }
    public void addButton(Component c, GridBagConstraints g,
                int x, int y) {
        g.gridx = x;
        g.gridy = y;
        g.gridwidth  = 1;
        g.gridheight = 1;
        add(c, g);
    }
}
```

〔練習題〕試應用 java.swing 類別程式庫的各 menu 選單類別合併圖 9.3 所示視窗，撰寫出圖 9.5 畫面的程式，完成的 JMenuBar 選單物件 jmb(或其他的名稱)，用

```
gkf.setJMenuBar(jmb);
```

的敘述式，加到視窗中。

撰寫圖 9.5 畫面 JMenuBar 選單物件 jmb 的程式，參考 14.2.3 小節。

〔練習題〕試以想像定義一個 *XxxListener* 的介面和 XxxAdapter 類別，並應斟酌有意義的事件名稱和方法程式，且說明採用原因。

想像定義一個 *XxxListener* 的介面和 XxxAdapter 類別，起頭參考樣板：

```
public interface XxxListener extends EventListener {
    public void xxxAction(XxxEvent e);
}
public class XxxAdapter implements XxxListener {
    public void xxxAction(XxxEvent e) {}
}
```

〔練習題〕試行將 DocumentPagePainter 範例程式，增加一個建構程式之參數部份為：

```
public DocumentPagePainter(File file, Font f) {
      …
```

```
}
```

增加建構程式用作 overloading**，程式並列示意：**

```
…
public DocumentPagePainter(JTextArea text, Font f) {
      …
}
public DocumentPagePainter(File file, Font f) {
      …
}
```

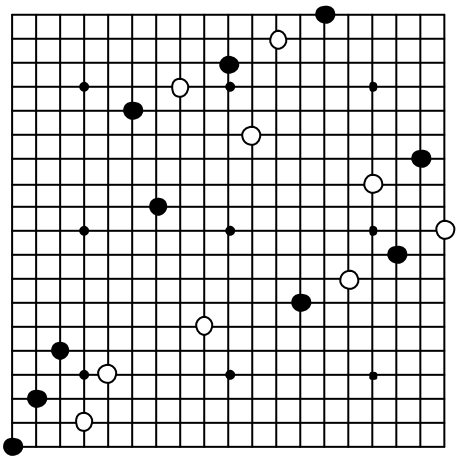
再試行用 `file` **方式來做設計**。

〔練習題〕文件列印作業可為一個獨立的線程，試將列印工作，用線程方式加入上述範例中。

將文件列印作業作為一個獨立的線程，可遵照第八章製作新線程或應用現有類別加上 implements Runnable **的方式**。

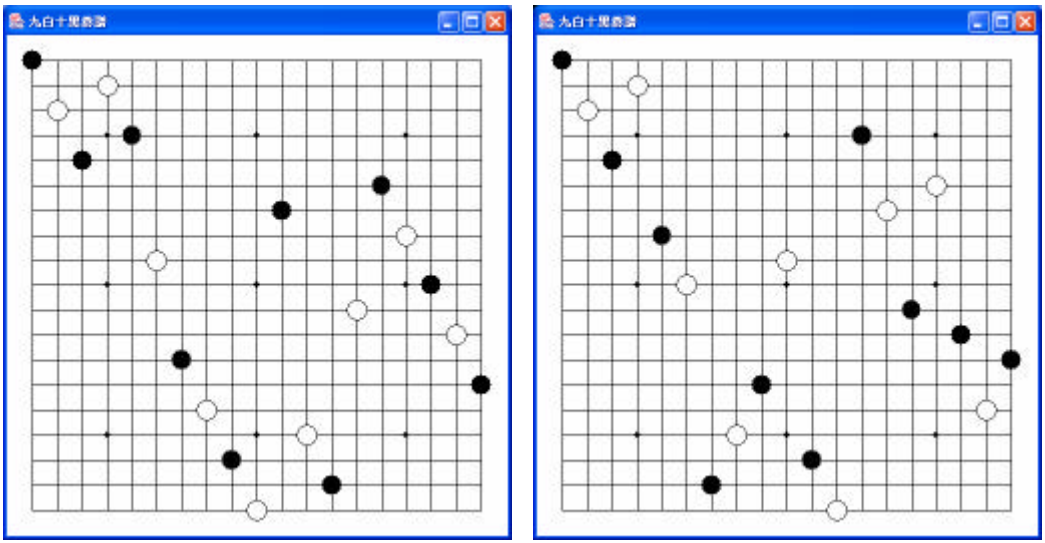
第十章

〔練習題〕「九白十黑碁位譜」，為在圍棋盤上顯示九顆白碁加上十顆黑碁的位置，無論在橫向、縱向或斜向的棋位上，都只能有一顆白碁或黑碁，不得有其他碁子。下圖為此種碁位譜的一個例子，試畫出此碁位譜圖。



(提示：可以找出擺棋的演算法，有助於畫出更多不同碁位譜的九白十黑。
[Griswold(1997)])

本「九白十黑碁譜」的命名只是一個隨興，而碁譜如未借助於電腦程式找碁位，則不易將碁擺出。下圖為此種碁位譜的另外兩個例子，但還有其他很多的例子，因此鼓勵運用演算法找出碁位，並應用本章說明的畫圖方式，逐子畫出。



```
import java.awt.*;
import java.awt.event.*;

class _9w10bQi {
    private int col;
    private int row;
    private _9w10bQi nearby;

    public _9w10bQi(int r, _9w10bQi q) {
        col = 1;
        row = r;
        nearby = q;
    }
    public boolean findMove() {
        while (nearby != null && nearby.canMove(col, row))
            if (nextMove() == false)
                return false;
```

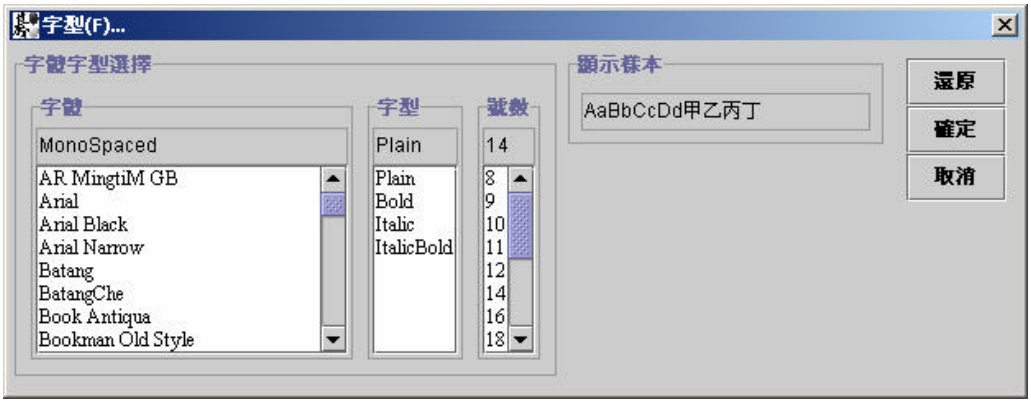
```
    return true;
}
private boolean canMove(int c, int r) {
    int rowDifference = r - row;
    if ((c == col - rowDifference) ||
        (c == col) ||
        (c == col + rowDifference))
        return true;
    if (nearby != null)
        return nearby.canMove(c, r);
    return false;
}
public boolean nextMove() {
    if (col < QiGame.board) {
        col++;
        return findMove();
    }
    if (nearby != null) {
        if (nearby.nextMove() == false ||
            nearby.findMove() == false)
            return false;
    }
    else
        return false;
    col = 1;
    return findMove();
}
public void paint(Graphics g) {
    int turn = 1;
    if (nearby != null) {
        turn = row%2;
        nearby.paint(g);
    }
    // 碁位在x,y交叉線上
    int grid = QiGame.gridsize;
    int qisize = grid * 4/5;
```

```
        int x = row * grid;
        int y = col * grid;
        if (turn == 1) {    // 畫黑子
            g.fillOval(x-qisize/2, y-qisize/2, qisize, qisize);
        } else {            // 畫白子
            g.drawOval(x-qisize/2, y-qisize/2, qisize, qisize);
            g.setColor(Color.white);
            g.fillOval(x-qisize/2+1, y-qisize/2+1,
                                          qisize-1, qisize-1);
            g.setColor(Color.black);
        }
    }
}

class QiGame extends Canvas {
    static int gridsize, board;
    private _9w10bQi Qi;
    private int ds, rs, boardsize;

    public QiGame(int b, int g) {
        board = b;
        gridsize = g;
        ds = gridsize/4;
        rs = gridsize/8;
        boardsize = b * g;
        for (int i=1; i <= board; i++) {
            Qi = new _9w10bQi(i, Qi);
            Qi.findMove();
        }
        enableEvents(AWTEvent.MOUSE_EVENT_MASK);
    }
    // 畫碁盤及碁子
    public void paint(Graphics g) {
      for (int i=1; i <= board; i++) {
        g.drawLine(gridsize*i, gridsize, gridsize*i, boardsize);
        g.drawLine(gridsize, gridsize*i, boardsize, gridsize*i);
```

```
        if (i==4 || i==10 || i==16) {   // 定石點位置
            g.fillOval(gridsize*i-rs, gridsize*4-rs, ds, ds);
            g.fillOval(gridsize*i-rs, gridsize*10-rs, ds, ds);
            g.fillOval(gridsize*i-rs, gridsize*16-rs, ds, ds);
        }
      }
      Qi.paint(g);
    }
    public void processMouseEvent(MouseEvent e) {
        if (e.getID() == MouseEvent.MOUSE_CLICKED) {
            Qi.nextMove();
            repaint();
        }
    }
}

public class QiPu extends Frame {
    public QiPu(int board, int grid, int framesize) {
        setTitle("九白十黑碁譜");
        setSize(framesize-grid, framesize);
        addWindowListener(new AppCloser());
        add(new QiGame(board, grid));
        setVisible(true);
    }
    public static void main(String[] args) {
        new QiPu(19, 25, 535);
    }
}
```

〔練習題〕以前述 ColorDialog 類別及 10.3 節所討論的字型 Font 類別為範本，試另設計一個獨立類別型式 FontSelector 的 Dialog 交談互動式選擇視窗，作為編輯檔案文字的字體、字型和文字號數大小的設定，顯示於如下圖顯示樣本的欄位

上，並能夠應用到第九章編輯程式，另外加入一個新的字型(F)點選項目。




```
package junitor;
import java.awt.*;

import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class FontSelector extends Jdialog
                            implements ActionListener {
  String[][] fontNamesStylesSizes;
  Font initFont, selectedFont;
  String textname;
  int textstyle;
  int textsize;
  int numOfList;
  JTextField nameTextField;
  JTextField styleTextField;
  JTextField sizeTextField;
```

```
JList[] list;
JTextField sample;

public FontSelector(JFrame jf, String s, boolean b, Font f) {
  super(jf, s, b);
  initFont  = f;
  getFontValues(f);
  setups();
  getContentPane().add(fontPanelLayout());
  setSize(624,240);
  addWindowListener(new FontDialogCloser());
}

void getFontValues(Font f) {
  textname  = f.getName();
  textstyle = f.getStyle();
  textsize  = f.getSize();
}

void setups() {
  GraphicsEnvironment ge =
          GraphicsEnvironment.getLocalGraphicsEnvironment();
  fontNamesStylesSizes = new String[][] {
      ge.getAvailableFontFamilyNames(),
      { "Plain", "Bold", "Italic", "ItalicBold"},
      { "8", "9", "10","11","12","14","16","18",
      "20","22","24","26","28","36","48","72"}
  };
  numOfList = fontNamesStylesSizes.length;
  list = new JList[numOfList];
  for (int i=0; i<numOfList; i++) {
      list[i] = new JList(fontNamesStylesSizes[i]);
      list[i].addMouseListener(new mouseListener());
  }
}
```

```
JPanel fontPanelLayout() {
  JPanel jp = new JPanel();
  jp.setLayout(new BoxLayout(jp, BoxLayout.X_AXIS));
  JPanel p1 = new JPanel();
  p1.add(fontFamily());
  jp.add(p1);
  JPanel p2 = new JPanel();
  p2.add(samplePanel());
  jp.add(p2);
  JPanel p3 = new JPanel();
  p3.add(buttons());
  jp.add(p3);
  return jp;
}

JPanel fontFamily() {
  JPanel jp = new JPanel();
  jp.setBorder(
          BorderFactory.createTitledBorder("字體字型選擇"));
  jp.add(fontNamePanel("字體"));
  jp.add(fontStylePanel("字型"));
  jp.add(fontSizePanel("號數"));
  return jp;
}

JPanel fontNamePanel(String s) {
  nameTextField = new JTextField(textname);
  return fontPanel(s, list[0], nameTextField);
}

JPanel fontStylePanel(String s) {
  styleTextField = new JTextField(fontStyle(textstyle));
  return fontPanel(s, list[1], styleTextField);
}

String fontStyle(int fs) {
```

```
  String style = null;
  switch (fs) {
    case Font.PLAIN:  style = "Plain"; break;
    case Font.BOLD:   style = "Bold"; break;
    case Font.ITALIC: style = "Italic"; break;
    case Font.BOLD+Font.ITALIC:   style = "ItalicBold"; break;
  }
  return style;
}

JPanel fontSizePanel(String s) {
  sizeTextField = new JTextField(Integer.toString(textsize));
  return fontPanel(s, list[2], sizeTextField);
}

JPanel fontPanel(String s, JList jl, JTextField textfield) {
  JPanel jp = new JPanel();
  jp.setBorder(BorderFactory.createTitledBorder(s));
  jp.setLayout(new BoxLayout(jp, BoxLayout.Y_AXIS));
  textfield.setEditable(false);
  jp.add(textfield);
  jl.setFont(new Font("Serif", Font.PLAIN, 12));
  jl.setFixedCellHeight(14);
  jp.add(new JScrollPane(jl));
  return jp;
}

JPanel samplePanel() {
  JPanel p = new JPanel();
  p.setBorder(BorderFactory.createTitledBorder("顯示樣本"));
  sample = new JTextField("AaBbCcDd甲乙丙丁", 16);
  sample.setEditable(false);
  p.add(sample);
  return p;
}
```

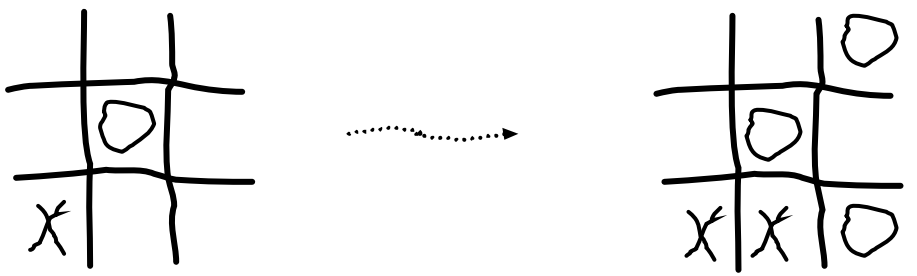
```
  private JPanel buttons() {
    JPanel p = new JPanel(new GridLayout(3,1));
    p.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
    p.add(makeButton("還原"));
    p.add(makeButton("確定"));
    p.add(makeButton("取消"));
    return p;
  }

  private JButton makeButton(String s) {
    JButton jb = new JButton(s);
    jb.addActionListener(this);
    return jb;
  }

  public void actionPerformed(ActionEvent e) {
    String s = e.getActionCommand();
    if (s.equals("取消")) {
        selectedFont = initFont;
        this.setVisible(false);
    } else if (s.equals("確定")) {
        this.setVisible(false);
    } else if (s.equals("還原")) {
        resetFont(initFont);
    }
  }

  private void resetFont(Font f) {
    getFontValues(f);
    nameTextField.setText(textname);
    styleTextField.setText(fontStyle(textstyle));
    sizeTextField.setText(Integer.toString(textsize));
    setAllTextFields(f);
  }

  private void setAllTextFields(Font f) {
```

```
  Font font = new Font(textname, textstyle, 12);
  nameTextField.setFont(font);
  styleTextField.setFont(font);
  sizeTextField.setFont(font);
  sample.setFont(f);
}

public Font getFont() {
  return selectedFont;
}

class FontDialogCloser extends WindowAdapter {
  public void windowClosing(WindowEvent e) {
      selectedFont = initFont;
  }
}

class mouseListener extends MouseAdapter {
  public void mouseClicked(MouseEvent e) {
    if (e.getClickCount() == 2) {
      for (int i=0; i<numOfList; i++) {
        if (list[i] == e.getSource()) {
          int index = list[i].locationToIndex(e.getPoint());
          if (index!= -1) {
            String s = fontNamesStylesSizes[i][index];
            switch (i) {
              case 0:
                  textname = s;
                  nameTextField.setText(s);
                  break;
              case 1:
                  textstyle = getFontStyle(s);
                  styleTextField.setText(s);
                  break;
              case 2:
                  textsize = Integer.parseInt(s);
```

```
                    sizeTextField.setText(s);
                    break;
                }
                selectedFont =
                          new Font(textname, textstyle, textsize);
                  setAllTextFields(selectedFont);
              }
              break;
            }
          }
        }
    }

    private int getFontStyle(String style) {
        int ts = Font.PLAIN;
        if (style.equals("Plain")) {
            ts = Font.PLAIN;
        } else if (style.equals("Bold")) {
            ts = Font.BOLD;
        } else if (style.equals("Italic")) {
            ts = Font.ITALIC;
        } else if (style.equals("ItalicBold")) {
            ts = Font.ITALIC+Font.BOLD;
        }
        return ts;
    }
  }
}
```

第十一章

〔練習題〕**九宮棋，亦有稱井字棋，乃因為棋的形狀如下圖示似中文"井"字而得稱，西洋則稱之這種棋為** TicTacToe**。九宮棋玩法規則簡單，開始時畫個井字棋盤，九個空格，然後對奕雙方輪流在空格內填入己方的棋子，同時要試著能夠先在橫向、縱向或者是斜向任何一個方向，下成都是己方的棋子，便為獲勝一方。本練習題製作人機對奕的九宮棋，不論是人先下還是機器先下，設計的程式雖可能不贏，但要絕為不可輸的『九宮不敗』小深藍棋手**(Little Deep Blue)。[莊(1998)]



九宮不敗的設計，觀念類似本章跳坑棋的 `intelliMove` **可以全盤搜尋法**(brute force)**方式，找出棋步。**

〔練習題〕**章節** 5.8.3 **的河內之塔，改寫為金環可用滑鼠移動的方式搬動，並設計成** Applet **的方式，可以掛到網站網頁內。**

河內之塔，改寫為金環可用滑鼠移動的方式搬動，參考 11.4 **節的跳坑棋。**

第十二章

〔練習題〕試將前 *WeekDays.html* 檔案 form 內的項目定義改用 select 元素標註另行製作該檔案，結果再做比較其差異。另外加進或改用其他屬性，又有些什麼不同(提示：可參考前面 select 元素標註章節說明)。

以 select/option 標註元素方式製作，可參考 12.4.2 小節。

〔練習題〕試以類似上述星球 Servlet 程式技術，應用於生肖、星座命理之類的網站。(12.4 節)

請先找出生肖、星座命理之類的網站，再參考 12.4.1 小節。

〔論述題〕試說明 compiled 與 scripting language 的差異，Servlet 與 CGI 的差異。

compiled 與 scripting language 的基本表像差異說明：

編譯型式 compiled 通指將原始程式轉為另一種主機碼或虛擬主機碼的格式檔案。直譯型式 scripting，也稱 command 或 interpreter，就直接執行原始直譯程式內所含的指令。

Servlet 是 Java 語言編譯型式的程式，CGI 是直譯型式的配合，相關 Servlet 與 CGI 的差異，參考 1.5.4 小節與本章前言部份。

第十三章

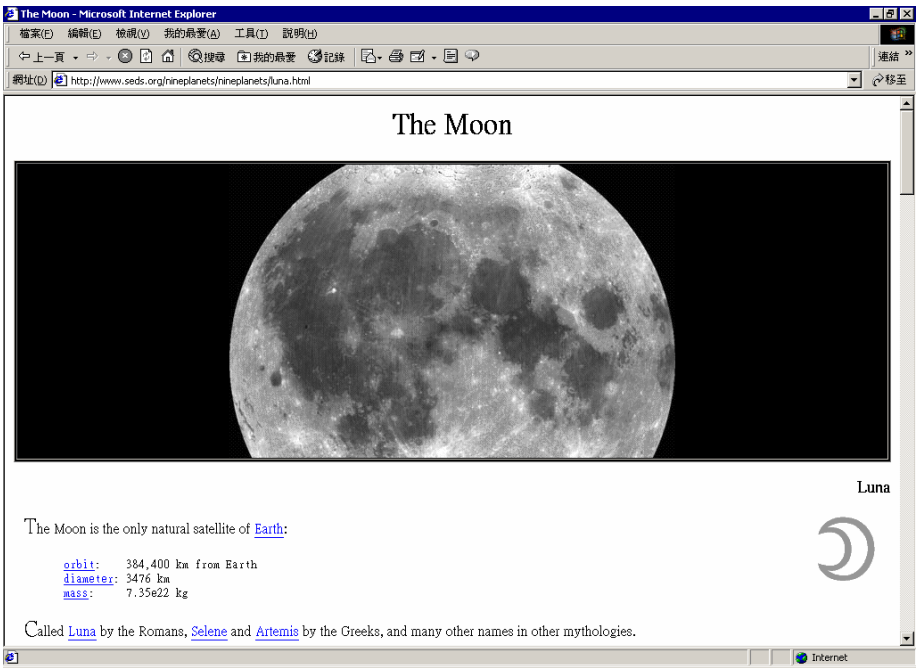
〔練習題〕本練習引用前章 Servlet 的 *WeekDays.html* 網頁，僅將 action 的 URL 改為 JSP 的程式如下：

```
<form action="/local/jsp/SolarDays.jsp" method="get">
```

其送出的網址，執行程式和查詢字串將像下式：

```
http://localhost:8080/local/jsp/SolarDays.jsp?Day=Monday
```

這個練習同時使用 JSP 網頁與 JavaBean 建構的 *SolarDays.jsp* 網頁程式，而依據前面查詢字串，本練習題將引到說明月球的網頁網站。



圖示採自 SEDS(Students for the Exploration and Development of Space)網站

將本節範例一 goodday.jsp 的編寫方式改為類似範例二採用 JSP 標準標註加上爪哇豆的編寫方式。

```
<%-- 本練習題主要在分隔 JSP 網頁與程式碼的示範 --%>
<!DOCTYPE html PUBLIC "-//W3C//DTD W3 HTML 4.0//EN">
<jsp:directive.page language="java"
        contentType="text/html"
        import="db.DayBean"
/>
<html>
<head><title>Good Day Greeting</title></head>
<body>
    <jsp:useBean id="dayBean" class="db.DayBean" />
    <center>
        <h1>Welcome</h1>
        <p>and</p>
        <h1>
        <jsp:getProperty name="dayBean" property="greeting" />
        </h1>
        <p>local time at server site is</p>
        <jsp:getProperty name="dayBean" property="localtime" />
    </center>
<jsp:include page="copyright.html" />
</body>
</html>
```

```
package db;
import java.util.*;
public class DayBean {
  private Calendar C = Calendar.getInstance();

  public String getGreeting() {
```

```
    int time = C.get(Calendar.AM_PM);
    return "Good " +
                ((time == Calendar.AM) ? "Morning" : "Afternoon");
  }
  public String getLocaltime() {
    return C.getTime().toString();
  }
}
```

參考解一：直接套用第十二章的 12.4.1 節 SolarDays.java 程式

```
<%@ page language="java" import="java.util.*"%>
<html><body>
<%! String echoString =
                "http://www.seds.org/nineplanets/nineplanets/";
    Hashtable planets = new Hashtable();
 %>
<%
    planets.put("Sunday",   "sol");
    planets.put("Monday",   "luna");
    planets.put("Tuesday",  "venus");
    planets.put("Wednesday","jupiter");
    planets.put("Thursday", "mercury");
    planets.put("Friday",   "mars");
    planets.put("Saturday", "saturn");
 %>
<% String DayString = request.getParameter("Day");
   if (DayString == null)
    out.println("There was no value selected");
   else {
    response.sendRedirect(echoString +
            planets.get(DayString) + ".html");
```

```
   }
 %>
</body></html>
```

參考解二：應用 JavaBean 方式

```
<!DOCTYPE html PUBLIC "-//W3C//DTD W3 HTML 4.0//EN">
<jsp:directive  page language="java"
                contentType="text/html"
                import="sdb.SolarBean"%>
<jsp:useBean id="SolarDayBean" class="sdb.SolarBean" />

<% String DayString = request.getParameter("Day"); %>
<%
  if (DayString == null)
    out.println("There was no value selected");
  else {
%>
<jsp:setProperty name="SolarDayBean" property="dayURL"
                value="<%= DayString %>" />
<%
    response.sendRedirect(SolarDayBean.getDayURL());
  }
%>
</body></html>
```

```
package sdb;
import java.util.Hashtable;
public class SolarBean {
  private String day;
  String echoString =
```

```
            "http://www.seds.org/nineplanets/nineplanets/";
  private Hashtable planets = new Hashtable();

  public SolarBean() {
    planets.put("Sunday",   "sol");
    planets.put("Monday",   "luna");
    planets.put("Tuesday",  "venus");
    planets.put("Wednesday","jupiter");
    planets.put("Thursday", "mercury");
    planets.put("Friday",   "mars");
    planets.put("Saturday", "saturn");
  }

  public String getDayURL() {
    return echoString + planets.get(day) + ".html";
  }
  public void setDayURL(String d) {
    day = d;
  }
}
```

第十四章

〔練習題〕試用 SQL 指令敘述式，找出表格中所列分數第二高的學員。

姓名	成績
張三	78
李四	81
王五	84
趙六	80

〔練習題〕再試找出及列示分數第三高以後所有的學員。

在本題 gradetable 表格及資料內容完成建立後，找出表格中所列分數第幾高的學員，可利用 ORDER BY score 建立一個經排序的暫時表格方式，然後以條件式取得任一名次的學員，條件式也可以設定為其他較複雜的形式，像用於取某一段的排名。(此參考題解系應用於 Oracle 資料庫系統)

```
SELECT name, score
FROM (SELECT name, score, rownum rank
      FROM (SELECT name, score FROM gradetable ORDER BY score DESC))
WHERE rank = 3
```

〔論述題〕試說明 JDBC 作資料庫連結的幾個重要步驟的觀念。

JDBC 作資料庫連結的幾個重要步驟，參考 14.2.3 小節

〔論述題〕試說明驅動程式和資料庫 URL 格式規則。

驅動程式 driver 的格式，參考 14.2.3 小節

資料庫 URL 的格式，參考 14.2.3 小節

Protocol:Subprotocol:DatabaseName

Protocol 用於起始資料庫連結，例如 jdbc 的 protocol。
Subprotocol 是資料庫系統驅動程式 driver 的名稱，例如 db2 驅動程式。
DatabaseName 為連結之資料源名稱，例如 sample 資料庫。

第十五章

〔論述題〕試說明 EJB Server 同時包括多個 EJB Container 容器的作用。

EJB Server 同時包括多個 EJB Container 容器的作用，參考 15.1, 15.3 節。

〔論述題〕試說明 JavaBean 技術和 EJB 技術的差異。

JavaBeans 技術和 EJB 技術的差異，參考 6.4 節與本章前言部份，並參考[Sun(2001) Enterprise JavaBeans Specification]和[Sun(1997) JavaBeans API Specification]。

〔論述題〕試說明企業豆介面的分類和差異。

企業豆介面的分類和差異，參考 15.4 節。

〔論述題〕試說明 Stateless 與 Stateful Session Bean 用途與使用時機上的差異。

Stateless 與 Stateful Session Bean 用途與使用時機上的差異，參考 15.7 節。

〔論述題〕試說明使用 Session Bean 與 Entity Bean 時機上的差異。

使用 Session Bean 與 Entity Bean 時機上的差異，參考 15.8 節。

〔**論述題**〕**試總合說明** Message-Driven Bean **訊息驅動式企業豆特性**。

總合說明 Message-Driven Bean **訊息驅動式企業豆特性，參考** 15.9 **節**。

第十六章

〔論述題〕試說明 HTML，XML，SGML 的差異以及個別設定的目標用途。

HTML，XML，SGML 的差異以及個別設定的目標用途，參考本章前言部份與 16.1 節。

〔練習題〕試依想像，試舉出本書所談到以外五種不同領域的 XML 標準，並加以簡單的說明。

隨意舉像 JavaML，EJBML，DBML，JSPML，MedML，GovML，…，AML，BML，CML，…，YML，ZML，…。三百六十行，行行是個無止境的想像與延伸，每種的制定，應有其專業的依據，以及潮流的符合。

〔思考題〕當百花齊放的 XML 標準產生時，讀者看官得如何因應著去想這問題？

百花齊放的 XML 標註語標準的產生，就像各類程式語言的發展，需求和創作是時勢和科技進步必然的現象，對於學習及應用者，可能會帶來一些負擔與壓力，然而確實掌握 XML 極大彈性的精神及技術，必較易於應變暨採納各種的標註語，且其專業的融入期也將縮短。因此如何由需求的學習，轉為應用的創作，並維持正面的態度，特別是在資訊科技領域裡，所經常要面對的挑戰。

〔論述題〕試說明何謂 DTD？

何謂 DTD 參考 16.2.4 及 16.2.8 小節。

〔論述題〕試說明 validating 與 non-validating 解譯程式的差異？又為何使用 non-validating 解譯程式？

參考 16.2.4 小節。

〔論述題〕使用 DOM 及 SAX 之差異？

使用 DOM 及 SAX 之差異，參考 16.2.9 小節。

〔論述題〕試舉五種表列固定(Enumerated)屬性值的元素屬性例子。

舉五種表列固定(Enumerated)屬性值的元素屬性例子參考

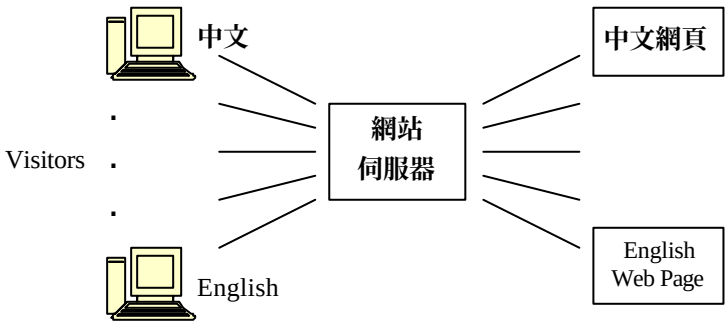
```
<!ATTLIST SPORTS game (basketball|football|baseball)>
<!ATTLIST ACADEMIC university (public|private)>
<!ATTLIST AUTOMOBILE energy (gasoline|diesel|electric)>
<!ATTLIST EMPLOYEE gender (female|male)>
<!ATTLIST VIDEO type (digital|analog)>
```

〔論述題〕試說明使用 CSS 樣規排版的一些優點。

CSS 樣規排版的一些優點，參考 16.3 節。

〔專題設計〕個人化網頁，可根據拜訪進來網站的屬性，而給予專屬化的網頁回

應，試用 XML 及 XSL 進行給予不同的網頁，例如根據 client 端所在語文作業的環境，進站後便自動給予相對的語文網頁，示意圖如後：



個人化網頁，參考 www.google.com，試用英文版和中文版視窗系統瀏覽程式檢測，可瞭解自動給予相對的語文網頁，對用戶的方便感受。

〔額外研究課題〕試將 *staff.xml* 文件利用圖 16.2，圖 16.3 的相關討論，轉換為 PDF 文件。

將 XML 文件轉換為 PDF 文件，參考 IBM:developerWorks:XML library。