雲平台與框架 API

--以 Google AppEngine 為例

1. 認識雲平台

1.1 以百貨公司比喻雲平台

傳統的網路服務是封閉型的 Client/Server 架構的延伸。其中的 Client 與 Server 兩端的軟件程序是各自開發的，Server 端開發人員撰寫完整的 Server 端程序，而 Client 端開發人員則撰寫完整的 Client 端程序並且調用 Server 端的程序。這種傳統的網路服務系統，其 Server 端如同一座四合院，庭院深鎖，外人(即 Client 程序)只能在大門口與四合院內的主人溝通。

如今的雲(Cloud)概念裡，其系統架構和軟件開發，大多來自傳統技術的延伸，並非特別的創新。然而它有了開放心懷，不再緊閉大門、深鎖庭院了，而是打開大門，提供庭院讓外人進來搭帳篷露營。此外，四合院主人還願意提供各項資源(軟硬件)和服務，甚至免付費呢!

上述的四合院比喻，還不能完整看出雲所帶來的商業模式和價值。開放型的四合院，也相當於目前大家常常去逛的百貨公司，如下圖：

圖 1　雲平台就像百貨公司

在四合院裡搭帳篷，也相當於在百貨公司裡開設專櫃做生意。例如，你到許多百貨公司處處可見到香奈兒、SK-II 等化妝品專櫃，如下圖：

圖 2　百貨公司裡的連鎖專櫃(企業)

基於雲的開放架構，Android 開發者能到各式各樣的雲(如 Google、Facebook 等)平台上，開發雲層上的應用程序(就如同進入別人的四合院裡搭帳篷露營、也如同到各個百貨公司開設連鎖專櫃)。於是，Android 應用服務就如同 SK-II 的國際連鎖服務。Android 手機端成爲連鎖店的營運總部，Android 手機也成爲連鎖店總經理的隨身指揮利器，和運籌帷幄中心了。

1.2　以連鎖專櫃比喻特定領域的(跨)雲平台

雲平台的分類

一般而言，有兩種常見的雲：

- **公有雲(Public Cloud)**
  ---- 開放給各行業或各領域的人們進來寫軟件或使用服務。
- **私有雲(Private Cloud)**
  ---- 並不開放給別人進來寫軟件或使用服務。

如果你學過 Java 或 C++的話，就知道類裡的函數可分爲：public、private 和 protected 共三種。如果對應到雲的分類上，將可以得到第三種雲，就是：

- **限定雲(Protected Cloud)**，即「特定領域的雲(Domain-Specific Cloud)」
  ---- 特定領域就是特定行業的雲，例如醫療行業、物聯網行業、保險行業、網路遊戲行業、KTV 行業等，各有其專屬(Protected)範圍。只開放熟悉該行業的人進來寫軟件或使用服務。

特定領域雲平台

特定領域的跨雲平台，其本身也是雲平台。所以「特定領域(跨)雲平台」又稱爲「特定領域雲平台」，例如，醫療領域雲平台的系統架構圖如下：

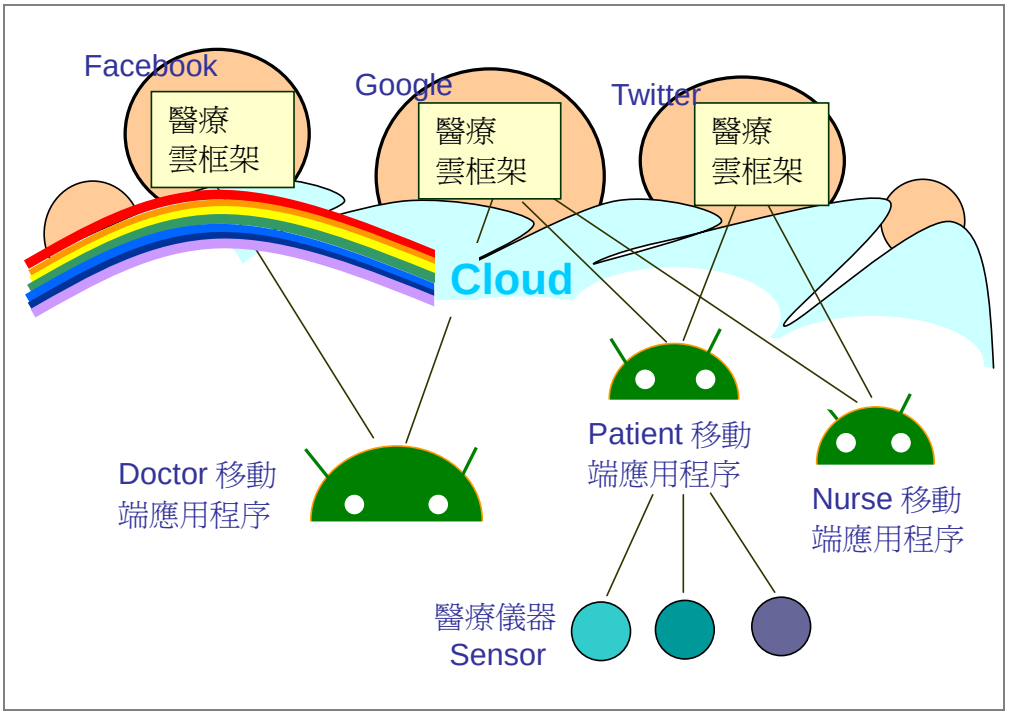


圖 3　醫療領域的(跨)雲平台

從圖可以看出來，這個醫療領域雲平台，是建立於多的公有(Public)雲平台上，甚至，可能也有自己的私有(Private)雲。所以它是一種跨雲(公有及私有)的雲。由於它是建立於公有雲的虛擬平台上，不需要巨大投資於硬件設備上，所以建置成本遠低於公有雲。欲理解這種新潮跨雲平台的最好比喻就是：

- 公有雲，如同「百貨公司」。
- 特定領域雲，如同百貨公司裡的 Chanel 連鎖專櫃。

大家都知道：

☆ 跨百貨公司的連鎖專櫃也是公司。

☆ 所以，跨公有雲的特定領域跨雲平台，也是雲。

2 認識 GAE(Google AppEngine)雲平台

GAE 的系統架構

GAE(Google AppEngine)是 Google 的雲服務引擎，第三方應用程序開發者能開發應用程序(AP)，然後放在 Google 伺服器上執行，不需擔心頻寬、系統負載、安全維護等問題，一切由 Google 代管。只要 AP 每月不超過 500 萬網頁面的流量就可享受免費的待遇。GAE 平台的系統架構如下圖：

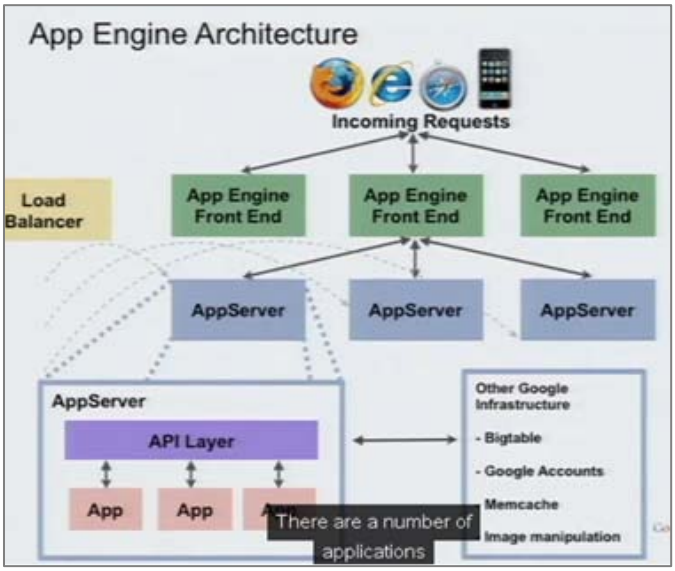


圖 4　GAE 雲平台架構(摘自 Google 公司文件)

從上圖可看到，從手機、PC、MID 等眾多端設備上，都能隨時上網發出要求(Incoming Requests)來存取 GAE 上的服務。在 GAE 後台的 AppServer 裡，GAE 提供了 API(即 API Layer)來銜接你的雲端應用程序(即 AP)。

3　GAE 範例：Slot Machine 小遊戲

水果盤拉霸機(Slot Machine，簡稱 SM)又稱爲老虎機、角子機或吃角子老虎機。如下圖：

圖 5　Android 拉霸機遊戲畫面

其玩法是先輸入投注金額(Bet)，然後拉動點擊把手或點擊<SPIN>鈕來轉動捲軸，捲軸會各自轉動，然後隨機出現不同圖案，如果停定時，有出現符合相同或特定相同圖案連線者，即依其賠率而勝出。同一家遊戲場裡的拉霸機通常會聯網，以投注額厘定累積大獎(Jackpot)金額，並隨時更新累績大獎金額，以便增加吸引性。

3.1 拉霸機遊戲軟件的設計圖

這遊戲軟件可分爲兩部分：

- 遊戲(Game)端部分，也就是 Android 手機端的應用程序。
- 櫃檯(Console)端部分，也就是 GAE 雲層 Servlet 程序。

當玩家押注後，按下<SPIN> 按鈕(開始加速滾動)，遊戲端就將「目前餘額」和「押注金額」傳送給 GAE 的櫃檯端程序。等待櫃檯端程序計算出中獎金額後，將「新餘額」和「獎項級別」回傳給 Android 遊戲端(滾動開始減速)，並更新遊戲端的畫面。其中，Android 遊戲端程序(ac01.java)發送 HTTP 來調用 GAE 雲層的 Servlet 接口，如下圖所示：

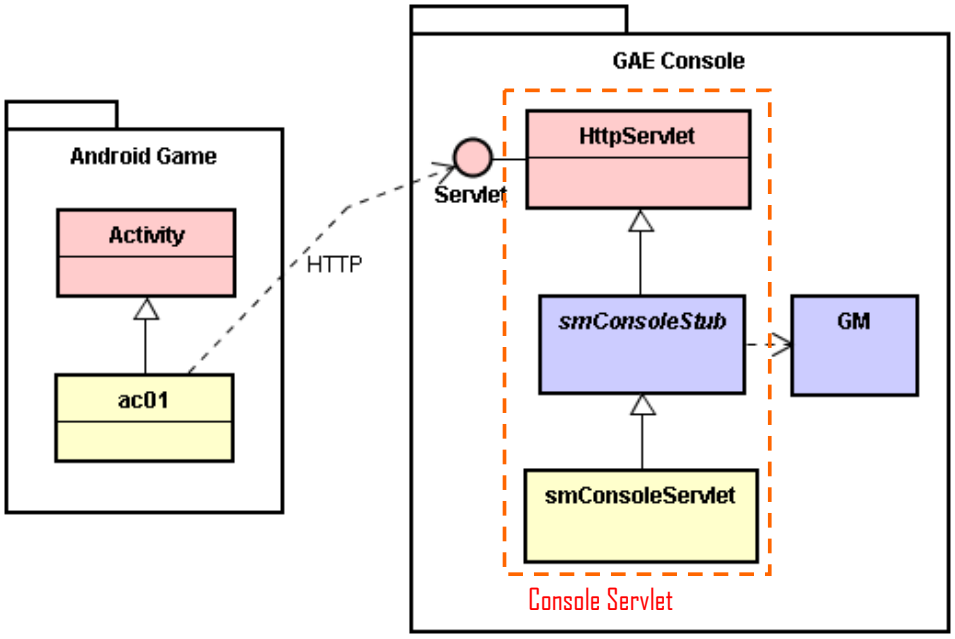


圖6 拉霸機遊戲的系統架構圖

Android 遊戲端透過 HTTP 和 Servlet 接口來傳送三種信息給 GAE 雲層。這三種信息爲：

- 當玩家啓動 Android 遊戲端時，發送"Init:"信息給 GAE 雲層程序。
  -- GAE 就從 DB 裡讀取玩家的餘額(即上回的餘額)，並回傳給遊戲端。
- 當玩家按下<SPIN>按鈕時，發送"Bett:*amount*,*bet*" 信息給 GAE 雲程序。
  -- 此信息附有餘額(amount)和押注金額(bet)，要求 GAE 程序決定「獎項級別(Rank)」，計算獎金和新餘額，然後回傳給遊戲端。

- 當玩家欲結束時，按下<Exit>按鈕發送"Fini:amount"信息給 GAE 雲層。
  -- 此信息附有目前餘額(amount)，GAE 接到信息，就依據將餘額存入 DB，完成時立即回覆給遊戲端，關閉遊戲端畫面。

GAE Console 端應用程序包含兩個部份：Servlet 模組和 GM 模組。GM(全名是 Game Machine)類是 Console 端應用程序的決策核心，例如決定遊戲獲獎的獎項，計算獎金等都是 GM 負責的任務。至於 Servlet 類體系則是負責與 Android 遊戲端的溝通任務。茲繪製 Console Servlet 模組的狀態機圖如下：

Slot Machine 的框架設計圖(方案一)

茲先設計(和開發)Slot Machine 應用框架，如下圖所示：

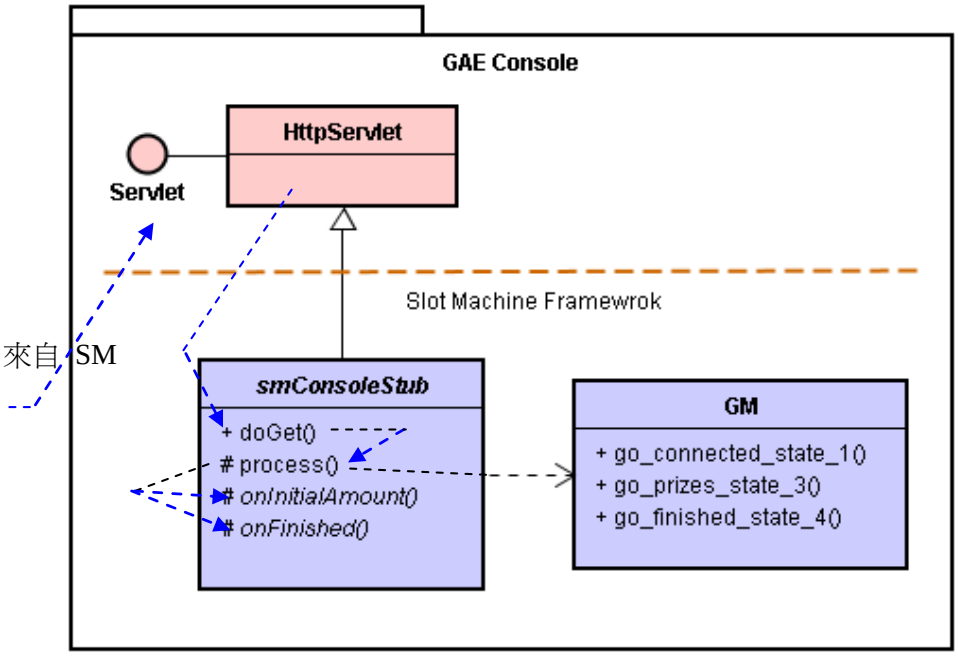


圖 7　Console 端遊戲框架設計圖(依據方案一)

當 Android 遊戲端(簡稱 SM)調用 HttpServlet 類的 Servlet 接口時，會轉而調用 smConsoleStub 類的 doGet()函數，此 doGet()轉而調用 process()函數去解析來自 Android 遊戲端的信息，然後調用 GM 類的函數，或調用應用程序的 onInitialAmount()和 onFinished()函數。

值得留意的是，此框架設計者(即 smConsoleStub 類設計者)決定了它與遊戲端溝通的信息格式(Format)，例如遊戲端必須使用"Init:"信息格式、"Bett:"信息格式和"Fini:"信息格式。一旦框架設計者決定了溝通接口，則應用程序開發者就必須遵循這些接口，而不能更改之。此時，框架就藉由這些接口來「框住」應用程序(及開發者)了。此外，框架也決定了它與子類間的接口，也就是決定了 onInitialAmount()和 onFinished()函數的名稱及參數格式。例如下圖：

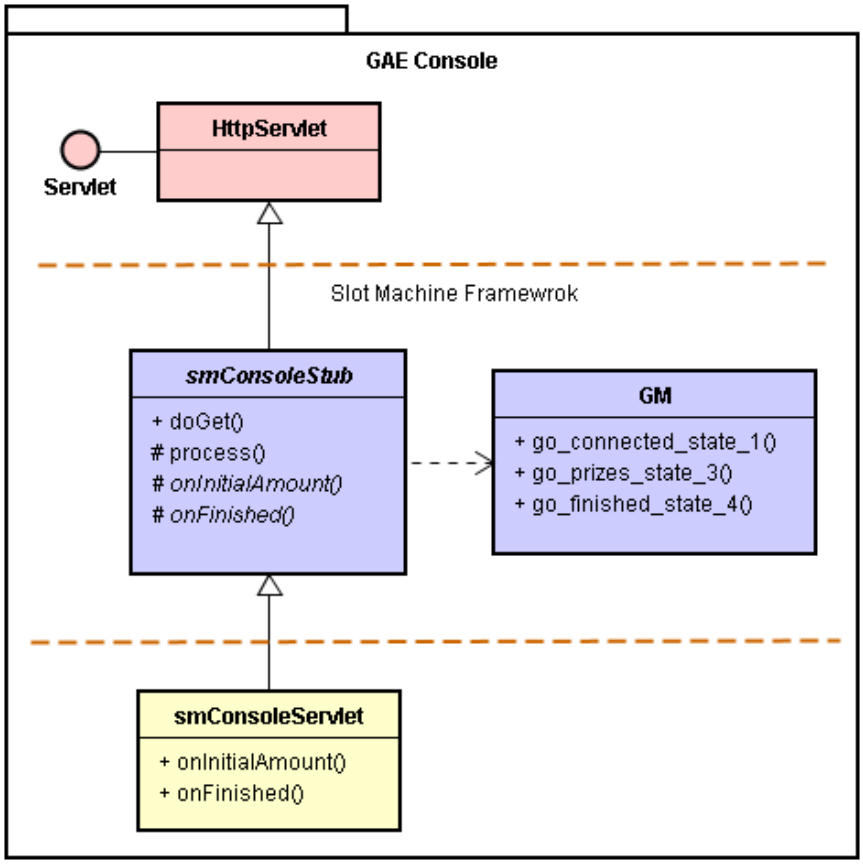


圖 8　Console 端的應用子類(smConsoleServlet)

其中，smConsoleServlet 子類就必須遵循 smConsoleStub 類的接口規定而實作(Implement) onInitialAmount()和 onFinished()兩個抽象函數。

Slot Machine 的框架設計圖(方案二)

在上述方案一裡，框架設計師決定了 Android 遊戲端與雲層之間溝通信息的格式，而應用程序開發者只能遵循之而不能制定自己喜歡的信息格式。如果想讓應用程序開發者能自行決定上述的信息格式，就可更改框架設計如下圖：

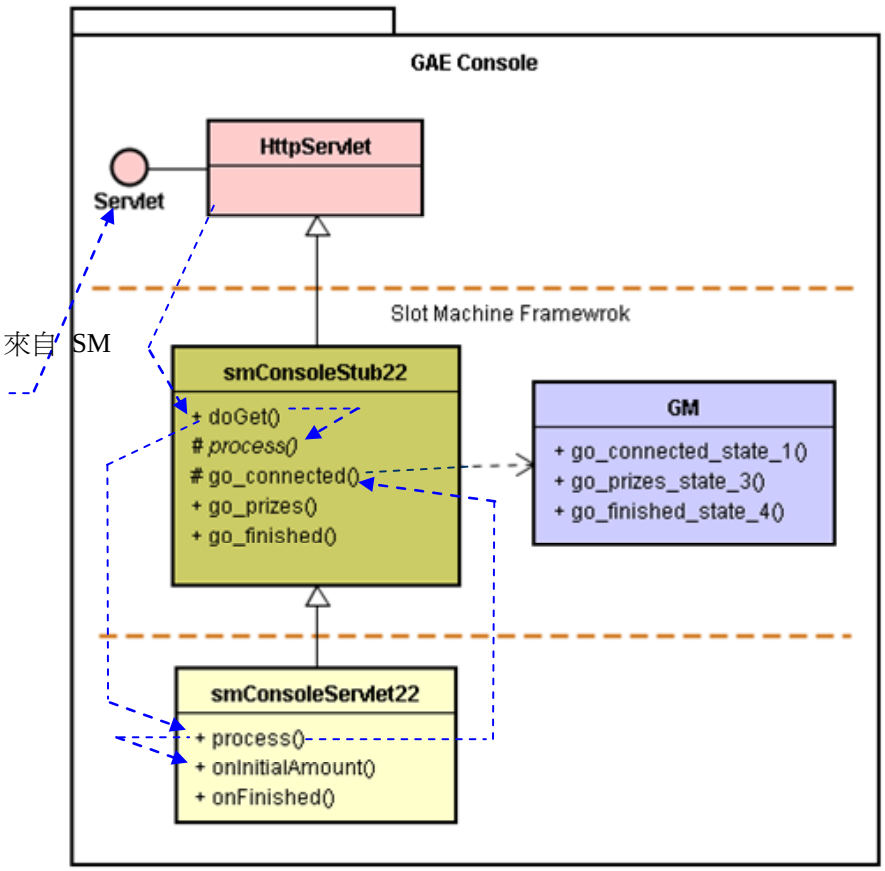
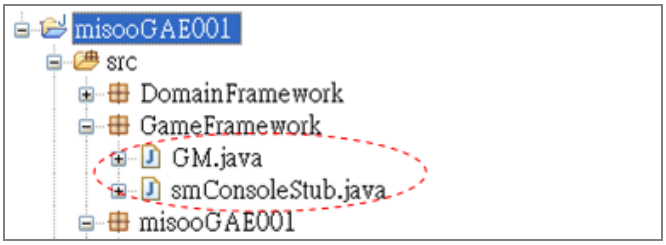


圖 9　Console 端遊戲框架設計圖(依據方案二)

在此新方案裡，smConsoleStub22 類的 process()是抽象函數，讓應用程序的 smConsoleServlet22 子類來實作之。框架裡的 smConsoleStub22 類只是將信息轉達給應用子類 smConsoleServlet22 而已，並不決定信息格式，也不解析信息。而是由 smConsoleServlet22 子類的 process()函數來解析信息。由於 Android 遊戲端的 ac01 類和雲層的 smConsoleServlet22 子類都屬於應用程序，由 ac01 類與 smConsoleServlet22 子類之間的溝通信息格式，是應用開發者可以自訂了。

3.2　撰寫 Console 遊戲框架的程序碼

這雲層框架是依據上述的方案一而設計的，其包含兩個類：GM 類和 smConsoleStub 類。茲在 GAE 開發專案裡定義上述類，如下圖：



茲撰寫 GM 類，其程序碼如下：

/*---- GM.java ----*/

```
package GameFramework;
import java.util.Random;

public class GM {
        public String state_var;
        public int current_amount = -1;;
        public int current_rank = -1;
        public Boolean status = null;
        //-----------------------------------------------
        public GM(){    this.go_state_0();    }
        public void go_state_0(){    state_var = "0";    }
        public void go_connected_state_1(int amt){
                if(! state_var.contains("0")){
                        status = false;
                        return;    }
                state_var = "1";
                current_amount = amt;
                this.go_state_2();
           }
        public void go_state_2(){
                    state_var = "2";
                    status = true;
           }
        public void go_prizes_state_3(int amt, int bet){
              if(! state_var.contains("2")){
                        status = false;      return;    }
              state_var = "3";
              // 計算獎金
              RC obj = new RC();
              current_rank = obj.getRandomInt(0, 1000);
              int prize = current_rank * bet;
              if(prize > 0) amt += bet;
              current_amount = amt + prize;
              this.go_state_2();
             }
       public void go_finished_state_4(){
                if(! state_var.contains("2")){
                        status = false;    return;    }
                state_var = "4";
                this.go_state_2();
           }
       public class RC{
             public int getRandomInt(int min,int max) {
                    try { Thread.sleep(2);
                    } catch (InterruptedException ex)    {    ex.printStackTrace();    }
                    Random randomizer = new Random(System.currentTimeMillis());
                    int k = randomizer.nextInt(max-min+1)+min;
                    if(k < 500) return 0;
                    if(k<750) return 1;
                    if(k<875) return 2;
                    if(k<938) return 3;
                    if(k<969) return 4;
                    if(k<984) return 5;
                    if(k<994) return 6;
                    if(k<1000) return 7;
                    return 0;    }
    }}
```

此 RC 類是依據隨機值而換算出獲獎的獎項(Rank)，其實各家遊戲場都有不一樣的獎項決定規則，而且隨時都可能更換新的獎項規則。上述 RC 類只是一個簡單範例而已。接著，茲撰寫 smConsoleStub 類，其程序碼如下：

/*---- smConsoleStub.java ----*/

```
package GameFramework;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

@SuppressWarnings("serial")
public abstract class smConsoleStub extends HttpServlet {
        private GM gm = null;
        private User user = null;
        private String strResult;
        protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                throws ServletException, IOException {
            UserService userService = UserServiceFactory.getUserService();
            user = userService.getCurrentUser();
            gm = new GM();    String gm_state = null;
            HttpSession session = req.getSession();
            Object obj = session.getAttribute("gmState");
            if(obj != null ){
                  gm_state = (String)obj;
                  gm.state_var = gm_state;    }
            String strCode = req.getParameter("code");
            String sv1 = req.getParameter("value1");
            String sv2 = req.getParameter("value2");
            int code = Integer.valueOf(strCode);
            if(code == 0) {
                    process(sv1);
                    if(gm.status == false) strResult = "Fail:99,99";    }
            else     strResult = Test(sv1, sv2);
            session.setAttribute("gmState", gm.state_var);
            resp.setContentType("text/plain");
            resp.getWriter().println(strResult);
      }
      private void process(String msg){
                  char cmd = msg.charAt(0);
                  if(cmd == 'I'){
                        gm.go_state_0();
                        gm.go_connected_state_1(onInitialAmount(user));
                        String strAmt = String.valueOf(gm.current_amount);
                        strResult = "Init:" + strAmt + ",99";
                       }
                  if(cmd == 'B'){
                        int idx = msg.indexOf(",");
                        String str_a = msg.substring(5, idx);
                        String str_b = msg.substring(idx+1);
                        int amt = Integer.parseInt(str_a);    int bet = Integer.parseInt(str_b);
                        gm.go_prizes_state_3(amt, bet);
                        String strAmt = String.valueOf(gm.current_amount);
```

```
                    strResult = "Bett:" + strAmt + "," + String.valueOf(gm.current_rank);
                   }
               if(cmd == 'F'){
                    int idx = msg.indexOf(",");
                    String str_amt = msg.substring(5, idx);
                    int amt = Integer.parseInt(str_amt);
                    boolean ret = onFinished(user, amt);
                    if(ret){
                         gm.go_finished_state_4();
                         strResult = "Fini:99,99";    }
                     else    strResult = "Fail:99,99";
          }}
     private String Test(String sv1, String sv2) {    return "***";     }
     abstract protected int onInitialAmount(User user);
     abstract protected boolean onFinished(User user, int final_amount);
}
```

Android 遊戲機端傳送 HTTP 信息給 GAE 雲層，就轉而調用上述的 doGet()函數。此時誕生一個 GM 對象，並從 session 取得"gmState"的值，並將此值存入 GM 對象裡，設定了 GM 對象的狀態值。接著，轉而調用 process()函數來解析信息內容，在依據內容而調用 GM 對象的函數或應用子類的函數，最後回傳信息給遊戲機端。

4 Android 框架 + GAE 框架

Android 的跨進程接口 IBinder，其角色相當於雲層裡的 Servlet 接口：

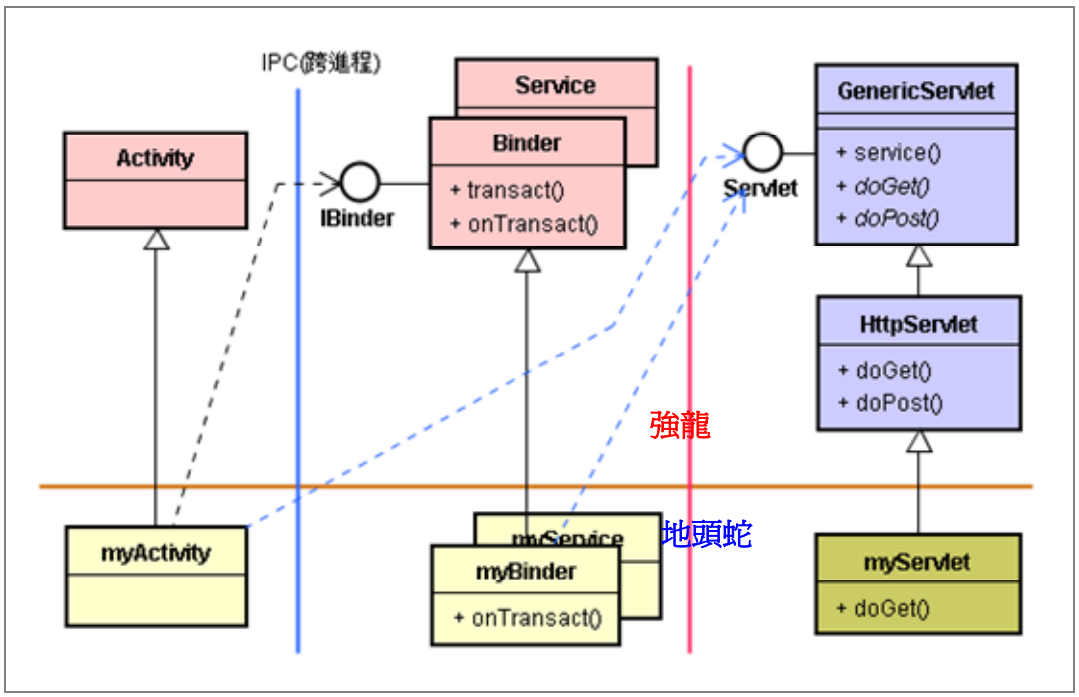


圖 10　Android 端框架與 GAE 雲框架的完美整合

無論 Android 還是 GAE 雲層都是以框架來支撐「強龍/地頭蛇」商業樣式。

茲寫個 GAE 框架範例，並進行架構設計如下：

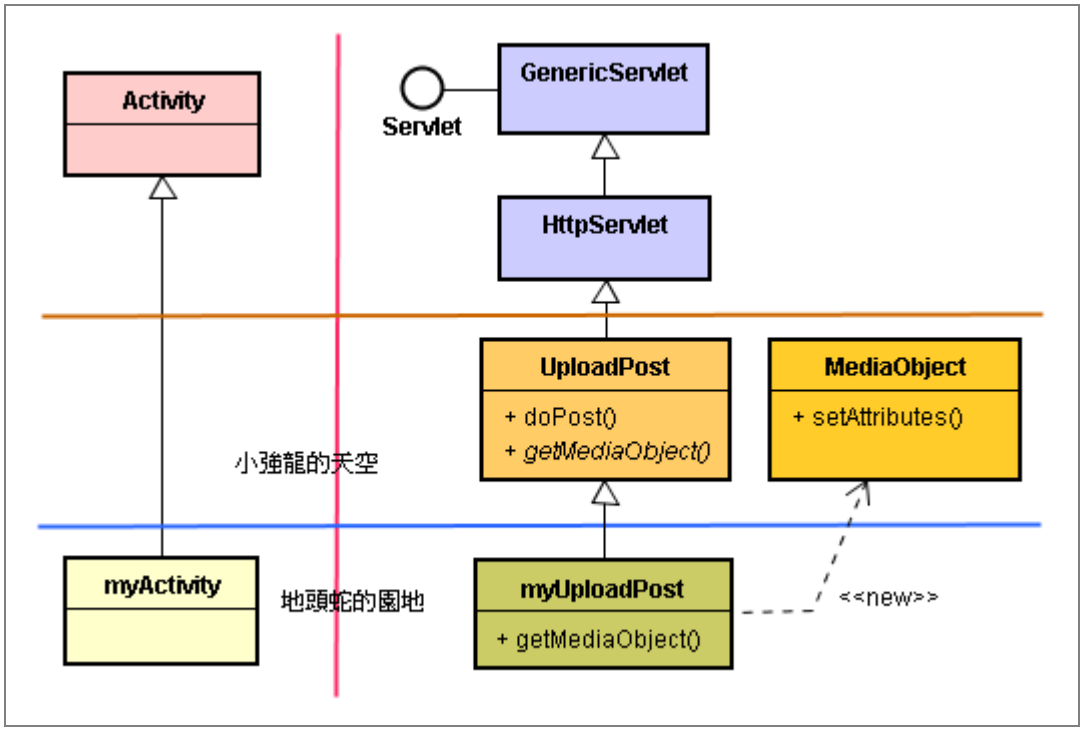


圖 11　Android 框架+ GAE 框架的範例

其中的 UploadPost 和 myUploadPost 兩個類的程序碼如下：

//UploadPost.java

```
package com.patrick.ccpmediastore;
import java.io.IOException;
import java.net.URLEncoder;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@SuppressWarnings("serial")
public abstract class UploadPost extends HttpServlet {

   public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
         IOException {

      try {
         PMF.get().getPersistenceManager().makePersistent(getMediaObject(req));
         resp.sendRedirect("/");
      } catch (Exception e) {
            resp.sendRedirect("/?error=" +
            URLEncoder.encode("Object save failed: " + e.getMessage(), "UTF-8"));
      }}
   protected abstract MediaObject getMediaObject(HttpServletRequest req) ;
}
```

// myUploadPost.java

```
package com.patrick.ccpmediastore;
import java.io.IOException;
import java.util.Date;
import java.util.Iterator;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.google.appengine.api.blobstore.*;
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

@SuppressWarnings("serial")
public class myUploadPost extends UploadPost {
   private BlobstoreService blobstoreService =
           BlobstoreServiceFactory.getBlobstoreService();
   protected MediaObject getMediaObject(HttpServletRequest req) {
             UserService userService = UserServiceFactory.getUserService();
             User user = userService.getCurrentUser();

             Map<String, BlobKey> blobs = blobstoreService.getUploadedBlobs(req);
             Iterator<String> names = blobs.keySet().iterator();
             String blobName = names.next();
             BlobKey blobKey = blobs.get(blobName);
             BlobInfoFactory blobInfoFactory = new BlobInfoFactory();
             BlobInfo blobInfo = blobInfoFactory.loadBlobInfo(blobKey);

             String contentType = blobInfo.getContentType();
             long size = blobInfo.getSize();
             Date creation = blobInfo.getCreation();
             String fileName = blobInfo.getFilename();
             String title = req.getParameter("title");
             String description = req.getParameter("description") + "from myNewUploadPost";
             boolean isShared = "public".equalsIgnoreCase(req.getParameter("share"));
              MediaObject mediaObj = new MediaObject(user, blobKey, creation,
                        contentType, fileName, size, title, description, isShared);
             return mediaObj ;
   }}
```

一樣的框架設計思維、方法和技術，應用於 Android 移動端上，同時應用於 GAE 雲層上。

5. 認識手機雲

-- 在 Android 平台上跑 i-Jetty web server

Jetty 是於 1996 年。是由澳洲 Mort Bay Consulting Limited 公司(由 Greg Wilkins 所創設的)開發的 Java-based web and application server and servlet container。Jetty 的第一代名稱爲 Mort Bay SERVLet servER，簡稱 MBServler。它是開源(open source)軟件。

其中，Mort Bay 是澳洲雪梨(Sydney)大城裡的一個地區的名字。這 Mort Bay 公司 Logo 圖像裡就是雪梨港口的照片，可以看到港口裡有一個小碼頭(jetty)，而且是 J 字母開頭的字，可呈現出它是 Java web server 的涵意，所以將 MBServler 改稱爲 Jetty。

由於 Jetty 一開始就設計成爲幕後型的 Java web server，可以嵌入於各式各樣的平台系統裡，所以大家其實已經天天在使用 Jetty 而不自知罷了。例如，許多赫赫有名的平台系統的幕後都使用 Jetty web server。例如 Eclipse、IBM Tivoli Netview、BEA WebLogic Event Server、Sybase EAServer、Apache Geronimo 等系統。

Jetty 設計得很精巧又穩定，它只需要很小的記憶裡空間，所以它適合運作於像 Android 等手持設備平台裡，例如 Android 手機、Android TV 等。於 2008 年，Android 版本的 Jetty 誕生了，稱爲 i-Jetty。一旦在 Android 手機、Android TV 等設備上執行 i-Jetty 時，可以讓更多人來與這些設備溝通、分享其資訊。未來，Jetty 也會在雲端服務(cloud service)平台上扮演重要角色，例如 2008 年，Jetty 在 Yahoo's Apache Hadoop clusters 平台裡扮演幕後主角，而締造了高度效率的歷史紀錄。

目前負責維護 Jetty 的是 Jetty community，其服務網頁是： www.mortbay.org。而 i-Jetty 的下載網頁是：http://code.google.com/p/i-jetty。如下圖：

圖 12　i-Jetty 的下載網頁

i-Jetty 本身是以 Android 應用程序形式嵌入(運行)於 Android 平台裡，它可以透過 Android 框架的 API 與其它應用程序溝通。因而，在 i-Jetty 裡執行的 Servlet 程序也能透過該 Android API 而與其它應用程序溝通、分享資料。如下圖 13 和圖 14：

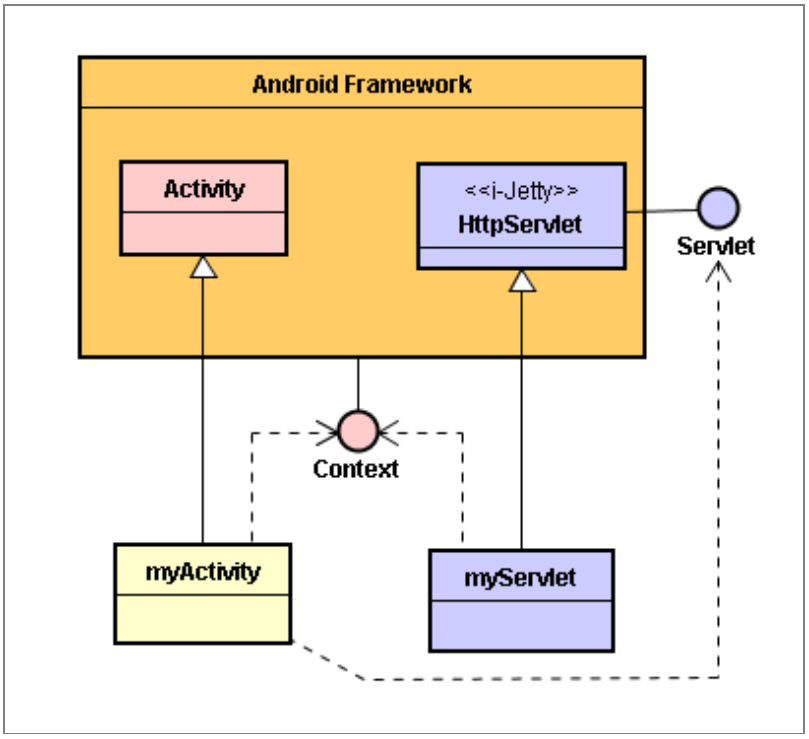


圖 13　單機(如 Android TV 或手機)內部的溝通

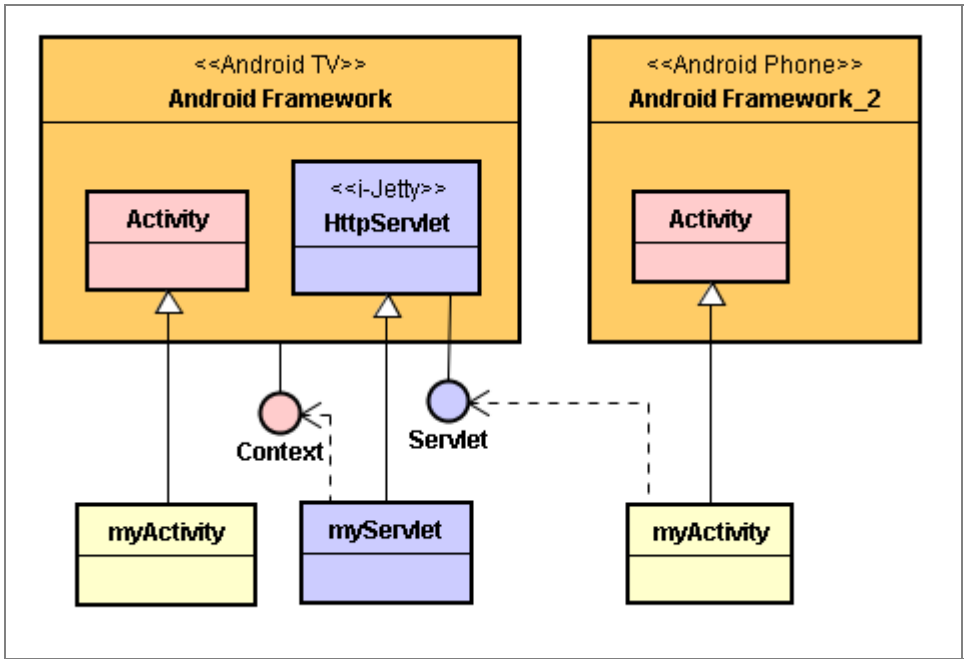


圖 14　跨機遠距(如 Android 手機與 Android TV 之間)溝通

Android 的一般應用程序透過 HTTP 協議的 Servlet 介面(接口)來與 i-Jetty 的 Servlet 溝通。而 i-Jetty 的 Servlet 則透過 Android 框架的 Context 介面來與 Android 應用程序取得聯繫，然後進行溝通。

在 myServlet 子類裡可利用 i-Jetty 提供的：

config.getServletContext().getAttribute("org.mortbay.ijetty.context");

指令來取得其當下的 Context 接口，如下述 i-Jetty 的範例程序碼：

```
// i-Jetty的servlet程序範例
//摘自i-Jetty套件的範例
import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/* ------------------------------------------------------------ */
/** Hello Servlet
 *
 */
public class HelloWorld extends HttpServlet
{
    String proofOfLife = null;

    /* ------------------------------------------------------------ */
    public void init(ServletConfig config) throws ServletException
    {
     super.init(config);
     //to demonstrate it is possible
        Object o = config.getServletContext().getAttribute("org.mortbay.ijetty.contentResolver");
        android.content.ContentResolver resolver = (android.content.ContentResolver)o;
        android.content.Context androidContext = (android.content.Context)config.getServletContext().getAttribute("org.mortbay.ijetty.context");
        proofOfLife = androidContext.getApplicationInfo().packageName;
    }

    /* ------------------------------------------------------------ */
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        doGet(request, response);
    }

    /* ------------------------------------------------------------ */
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        response.setContentType("text/html");
        ServletOutputStream out = response.getOutputStream();
```

```
            out.println("<html>");
            out.println("<h1>Hello From Servlet Land!</h1>");
            out.println("Brought to you by: "+proofOfLife);
            out.println("</html>");
            out.flush();
        }
}
```

上圖裡是從同一支手機(本機)裡的 myActivity 來呼叫本機的 Servlet 接口，這是可行的，但是意義比較小。更大的意義是：由別的手機裡的 myActivity 來傳送 HTTP 呼叫到本機的 Servlet 接口，呼叫到本機的 myServlet 子類；然後由 myServlet 呼叫 Android 的 Context 接口，而完成兩支手機裡兩個 myActivity 子類之間的相互溝通。◆