

# 目錄

寫在最前	3
第 1 天: HelloWorld	4
第 2 天: 會記名字的 HelloWorld	13
第 3 天: 預定時間的HelloWorld	20
第 4 天: HelloWorld 日誌	27
第 5 天: HelloWorld 日誌 +	33
備忘	37
<i>Eclipse</i> 內打中文	37
除錯指南	38
<i>Novcom Not Running</i>	38
更正內容	39
<i>Listener</i> 正確的設置方法	39
處理彈出的 <i>Alert Dialog Box</i>	39
有用連結	40
官方文檔 ( <i>SDK</i> 相關)	40
<i>SDK</i> 相關 (非官方資料)	40
討論區收藏	40
基礎知識	40

除錯相關

40

版權聲明

41

# 寫在最前

這個學習的筆記是個人學習 Mojo SDK 的一些經驗分享, 不是什麼正式的教學, 不過有興趣的朋友亦可以當教學使用, 亦不是一些專業的書籍, 不過大家都可以當作 Mojo SDK 學習的參考. 這筆記會給所有愛 WebOS, 想學 Mojo SDK 的人免費參考及閱讀, 不過所有內容都是有版權保護的, 如需要轉載, 請必須寫明出處, 即是我的Blog:

Prē-Dict:

<http://www.pre-dict.blogspot.com>

Prē-Dict Twitter:

[http://twitter.com/pre\\_dict](http://twitter.com/pre_dict)

如果有興趣交流, 或給意見的話, 可以有以下聯絡方式:

E-mail / Facebook:

[ted.wong.0430@gmail.com](mailto:ted.wong.0430@gmail.com)

Twitter:

<http://twitter.com/tedwong0430>

另外, 我希望我的學習筆記可以幫到更多學習 Mojo SDK 程式的人, 所以我會不斷更新最多的資料, 不過因為寫這筆記的準備不足, 所以還得大家見諒, 有很多基本上的概念都是欠奉的. 大家可能要等等, 或者自己先上網找資料.

# 第 1 天: HelloWorld

因為這是一本學習筆記, 所以我不會有一些很有系統的安排, 只是隨心而寫, 而多數第一個學習的程式, 第一個往往都是 HelloWorld, 那我就先做一個 HelloWorld.

這是我最初的想法: 這是一個程式, 給我輸入名字, 便按一個鍵, 再叫出一個 Dialog 向我說一句 Hello.

(有關如何設定Mojo SDK 的開發環境, 我會最後先寫, 現先假定所有東西都準備好了)

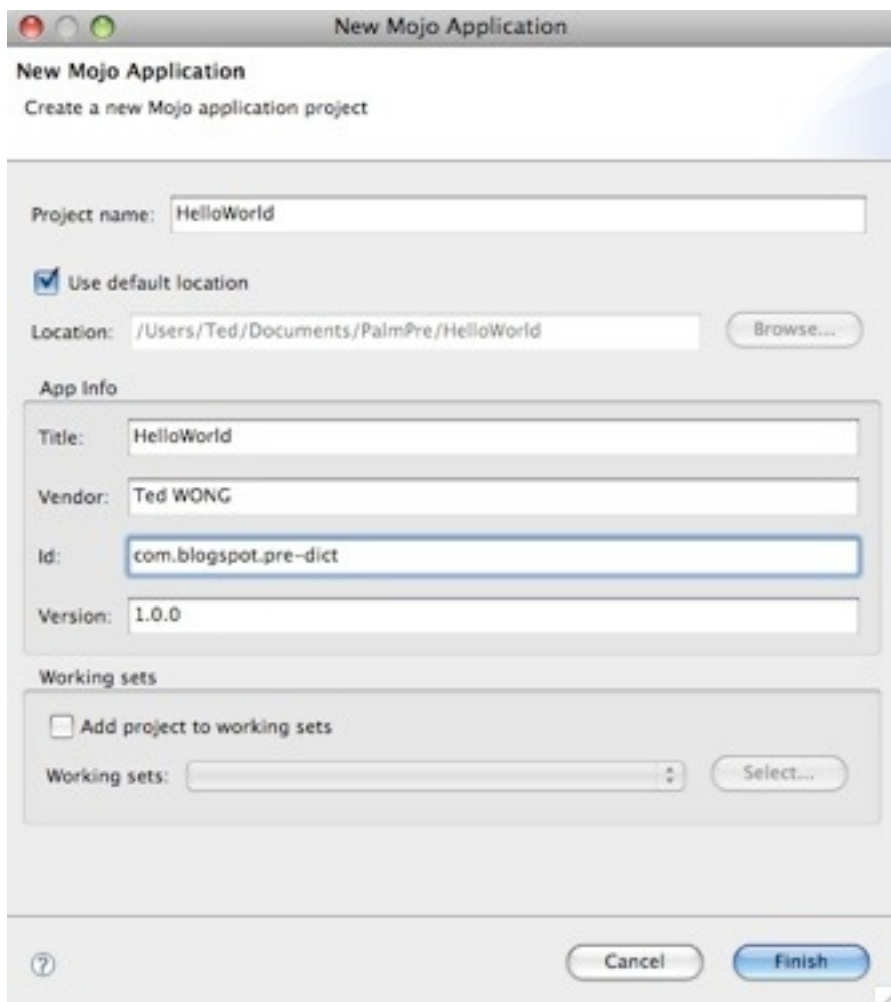
- 步驟 1:

開啟 Eclipse

File > New Mojo Application

- 步驟 2:

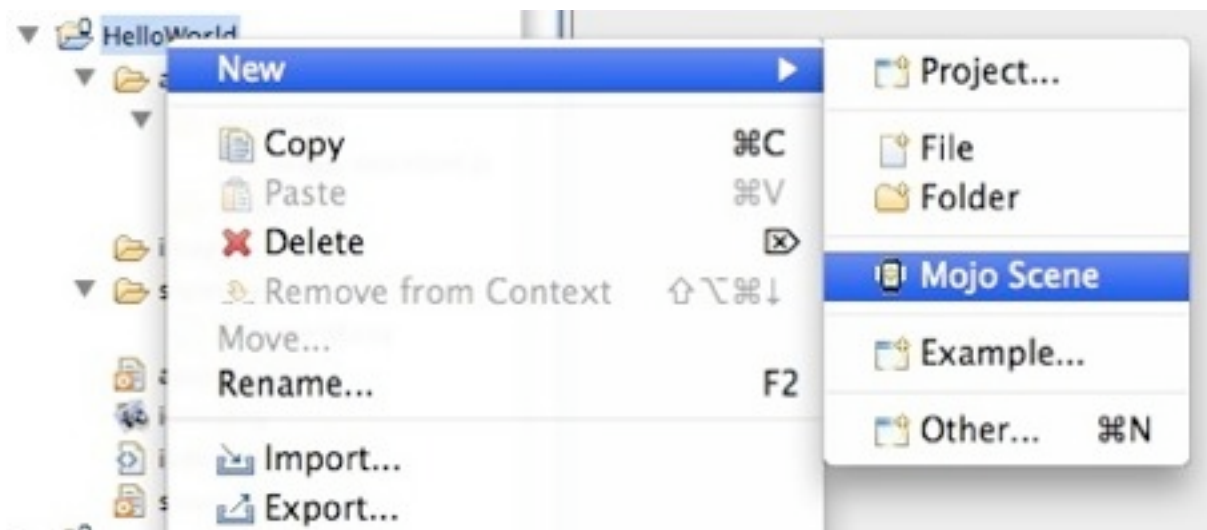
輸入以下資料 > 之後按 "Finish"



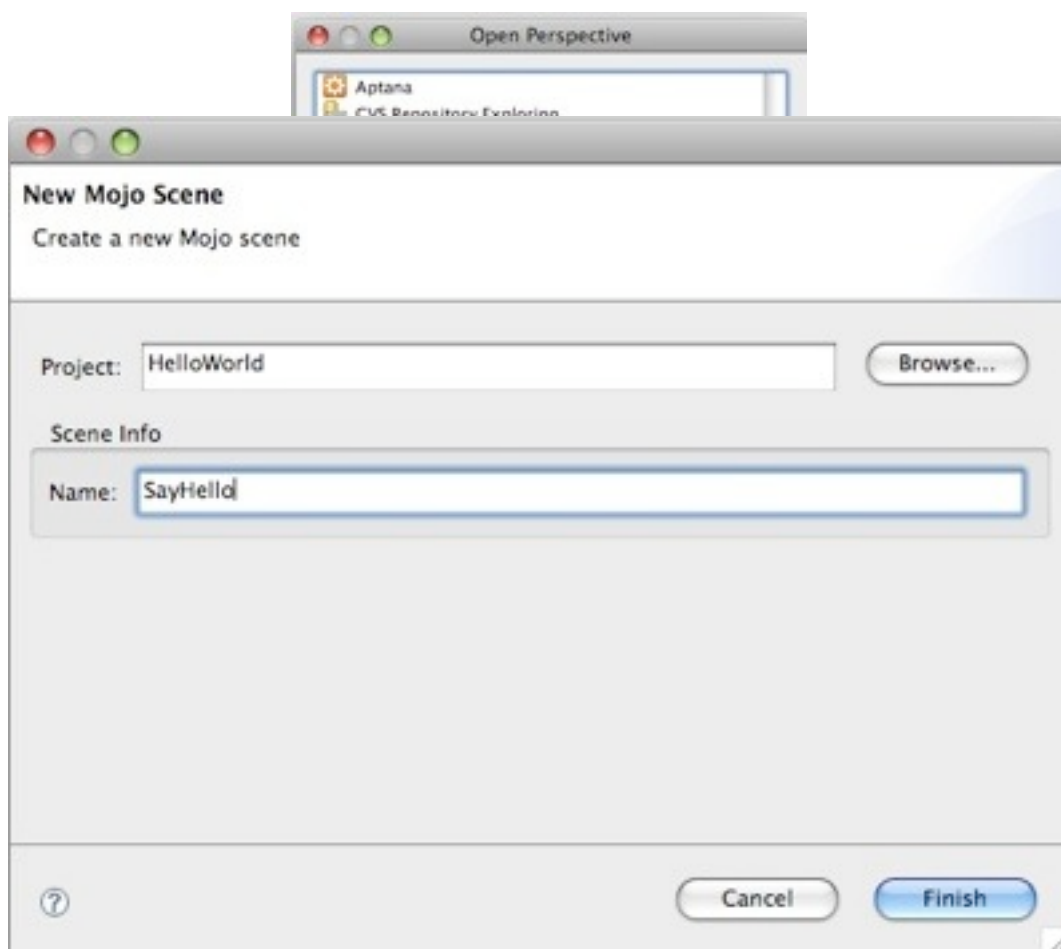
最後更新日期: 01/08/2009

• 步驟 3:

Eclipse



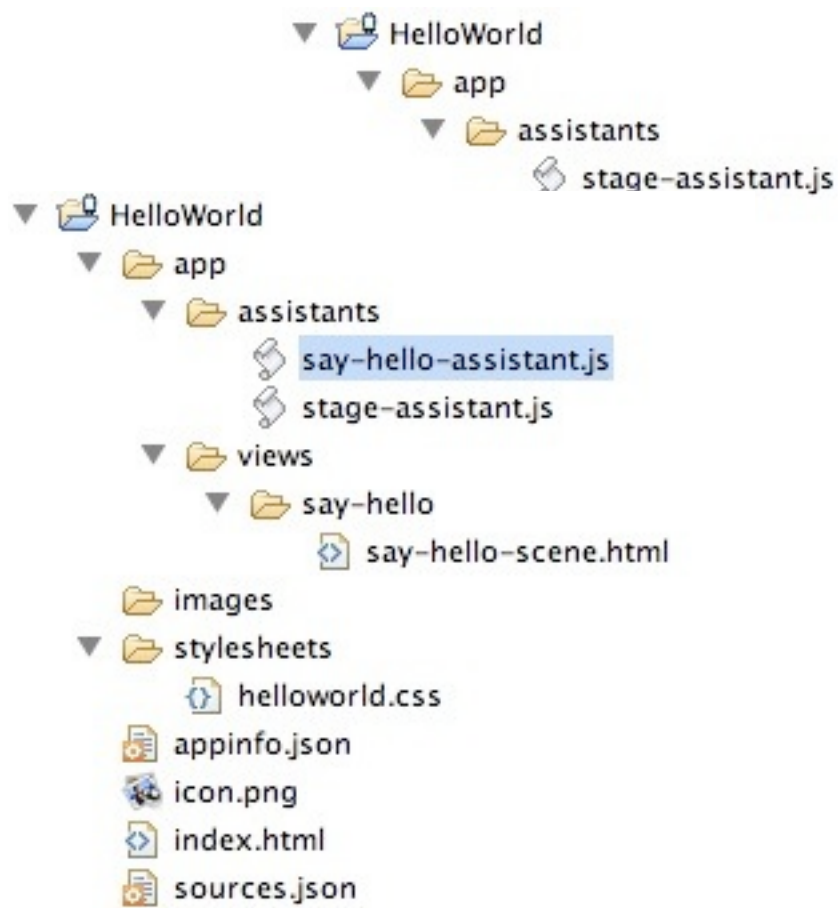
Windows > Open Perspective > Others > 選擇 webOS > OK



在

Explorer 中看到HelloWorld 的骨架已經做好:

Project



最後更新日期: 01/08/2009

- 步驟 4:

在 Project Explorer 中的HelloWorld Folder 右手鍵 > New > Mojo Scene

- 步驟 5:

輸入以下資料, 做一個新的Scene, 名為 SayHello

最後更新日期: 01/08/2009

當完成了之後, 看看Project Explorer

發現Eclipse 已幫助做了一堆檔案, 分別是 say-hello-assistant.js

say-hello 的文件夾, say-hello-scene.html

如果有細心留意, 在 sources.json 入面都會多了一個東西:

```
{
  "scenes": "say-hello",
  "source": "app\\assistants\\say-hello-assistant.js"
}
```

我們做了一個Scene, 是用來sayHello 用的.

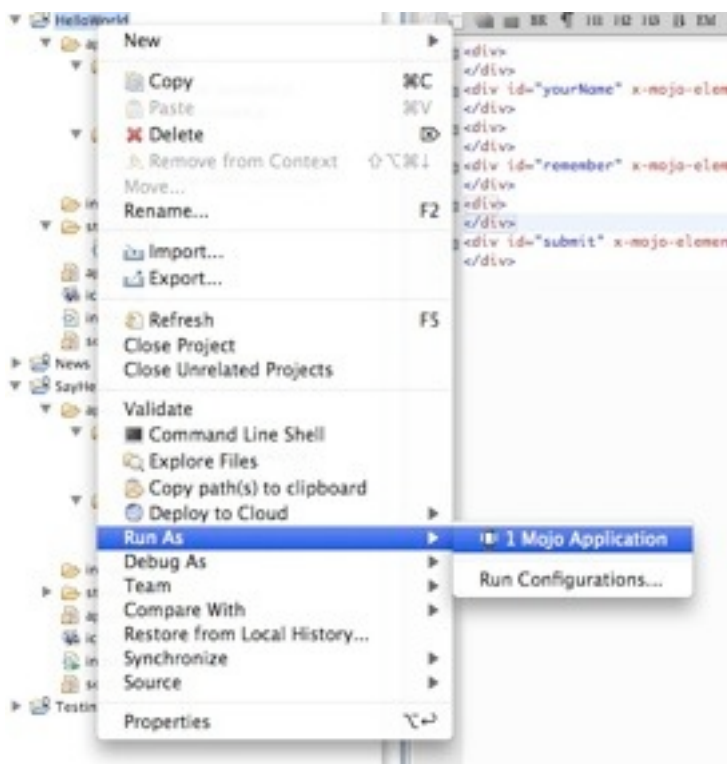
say-hello-assistant.js 是用來放置一些程式設置的東西.

say-hello-scene.html 是這個scene 的UI 鋪排用的.

#### • 步驟 6:

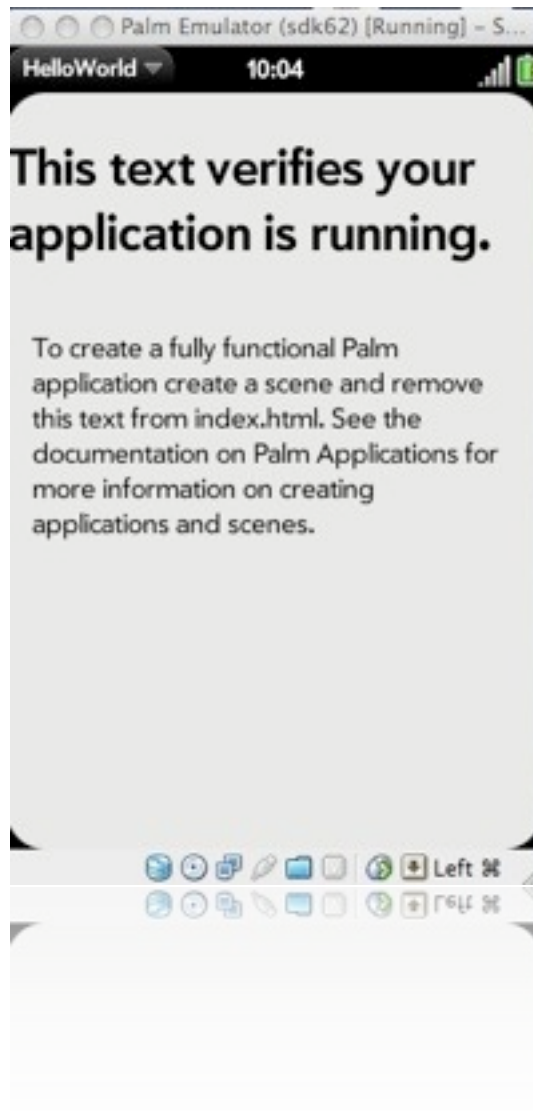
先開了Palm Simulator

再試試一切東西都是正式運作, 在 Project Explorer 中的HelloWorld Folder 右手鍵 > Run As > Mojo Application



在 WebOS 的Emulator, 應該可以看到:





- 步驟 7:

因為在WebOS 上, 每一塊畫面, 都叫做Scene, 就像網頁的一頁, 由於第一個是index, 而第一個assistant 會是stage-assistant, 而第一個叫的method 會 `StageAssistant.prototype.setup()`, 我們不想看到第一頁的index, 除了將他修改之外, 我們可以嘗試轉去第二個Scene, 而這我們的例子所做, 我們做到的第一個Scene 會是 say-hello, 那我們就改寫這個method, 呼叫controller 來將say-hello 的scene 變成我們現在的scene, 將 `stage-assistant.js` 的method 改成如下:

```
StageAssistant.prototype.setup = function() {  
    this.controller.pushScene("say-hello");  
}
```

- 步驟 8:

現在要設定UI, 而UI 的做法是 HTML + CSS, 非常簡單, 先將say-hello-scene.html 改成以下的Source Code

```
<div id="yourName" x-mojo-element="TextField">  
</div>  
<div>  
</div>  
<div id="submit" x-mojo-element="Button">  
</div>
```

<http://www.pre-dict.blogspot.com>

你會看到很多叫做 x-mojo-element 的東西, 這是說明這會是什麼的 webOS widget 會在這裡, 你可以發現我分了三大部份, 3個 div tag, 其中一個是Textfield, 供輸入名字用, id 叫做 yourName, 還有一個button, id 叫做submit.

id 是非常重要的, 不可以重疊, 而且要易記, 易明, 因為你後面很多機會會用到.

- 步驟 9:

看say-hello-assistant.js 的時間, 你會發現這已替你做好很多功夫, 有setup, activate, deactivate 和cleanup 這四個method 已做好, 不過如何決定那一個呢, 暫不詳細討論, 先搞清楚第一個會載入的. 進入Scene 時, 會第一時間呼叫setup, 而widget 的setup, 和 listener 的設置都是這裡處理的.

那我們改一改原先的setup method 變成如下:

```
SayHelloAssistant.prototype.setup = function(){
    var attributes = { //設定一個attributes
        hintText: $L('Your Name ...'), /*如果沒有輸入文字，就有一個提示的字樣給你看要打什麼東西*/
    };

    this.model = {
        value: "", //原本Textfield 的字
        disabled: false //可以修改
    };

    /*將id "yourName" 的 widget 套用attributes 和 model 的內容都放進去,
    * 設定為一個mojo text field
    */
    this.controller.setupWidget("yourName", attributes, this.model);

    /*這句和之前很似，不過這句是一句寫完，
    *找一個id 為 "submit" 的 widget 套用 this.attributes,
    * 而 this.attributes 內是空的，即使所有用預設的，而在model 上，
    * 則將button的 Label 變成submit
    * */
    this.controller.setupWidget("submit", this.attributes = {}, this.model = {
        buttonLabel: "submit"
    });

    /* 呼叫controller listen id 為submit 的widget,
    * 而listen 的動作是 tap,
    * 當listen 到有關的動作是，呼叫sayHello
    * */
    this.controller.listen("submit", Mojo.Event.tap, this.sayHello.bind(this));
}
```



#### • 步驟 10:

因為我們答應了Listener 當submit 鍵按的時間，便會叫一個sayHello 的method 來應付使用者的按鍵：

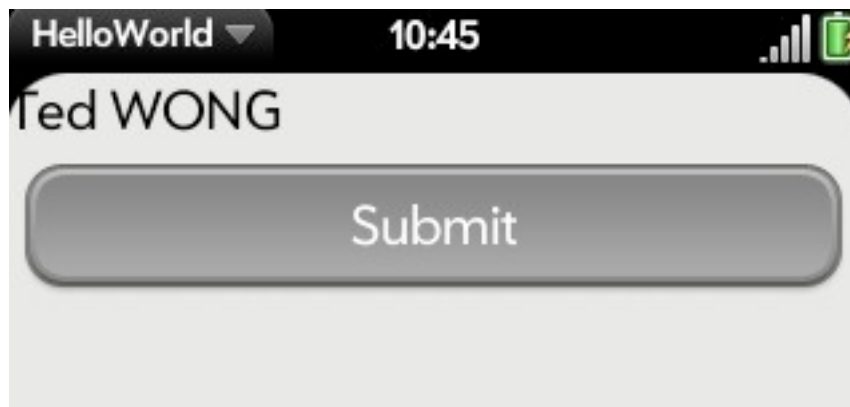
```
SayHelloAssistant.prototype.sayHello = function(){
    /*彈出一個errorDialog, 上面寫 Hello,
    之後將 id 為 yourName 的mojo widget 的 value 拿出來
    並寫在"Hello "之後.
    */
    Mojo.Controller.errorDialog("Hello " + this.controller.get("yourName").mojoWidget.value());
};
```

#### • 步驟 11:

當我們再run 一次的時間，便發現這個程式已經運作，不過

1. 當 yourName 沒有輸入時，submit 鍵便會縮上去
2. yourName 頂得太高，不美觀。





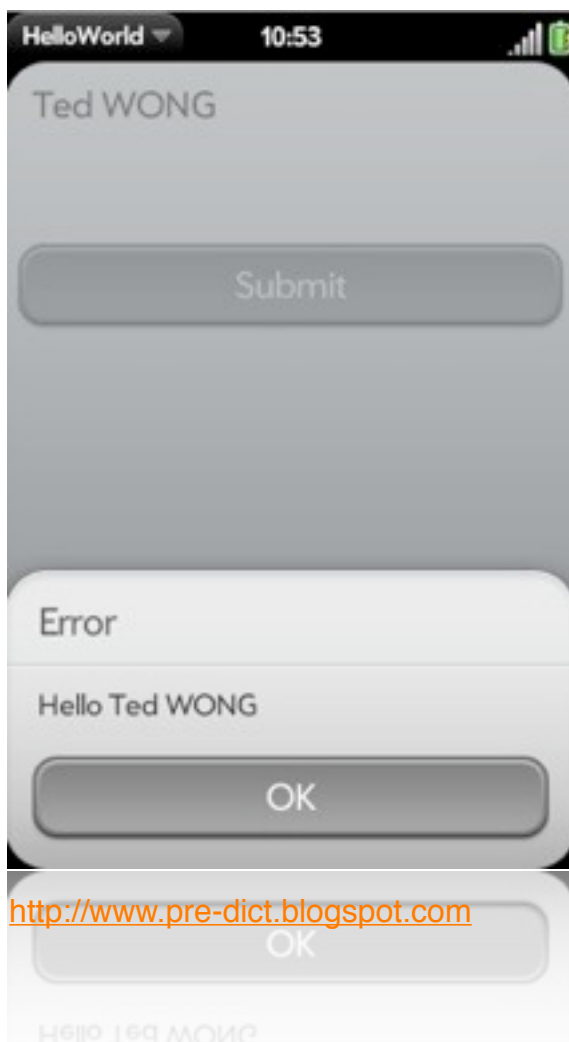
要解決這個問題可以用CSS 解決, Palm 其實已經準備了一堆CSS 給我們, 當然我們可以自己做, 不過第一天, 沒有這個心情雅興, 所以還是借用Palm 的. 將say-hello-scene.html 變成如下:

```
<!--palm-text-wrapper, 令文字更易看, 不會貼近邊位-->
<div id="yourName" x-mojo-element="TextField" class="palm-text-wrapper">
</div>

<!--中間放置一個spacer,
避免submit button 撞到 yourName text field-->
<div class="palm-header-spacer">
</div>

<div id="submit" x-mojo-element="Button">
</div>
```

今天學到的就是這些, 明天再努力探索:



## 第 2 天: 會記名字的 HelloWorld

我打算這個星期都是做HelloWorld, 一直不停地, 無止境地將我這個 HelloWorld 變成地上最強的 WebOS HelloWorld, 如果有跟我筆記的朋友, 請不要刪除你的舊檔案, 因為這個 HelloWorld, 是建基於之前的HelloWorld.

今次的HelloWorld 會用到Toggle Button 和 Cookies. Cookies 就是網頁一向用開的Cookies, 記低一些使用者的資料, 方便使用者在瀏覽時有更好的體驗. (亦方便廣告商如Google 按你的使用習慣來為你賣廣告) Toggle Button 是一個新的東西, 在iPhone 之前, 未有這個東西, iPhone 如果處理on/off, T/F 的抉擇時, 不會要你在Combo Box 上打剔, 而是很方便地向左/右滑行. 這個是一個很型的動作, 亦是較方便給手指操作的. WebOS 都學了這個功能, 今次我都會利用到.

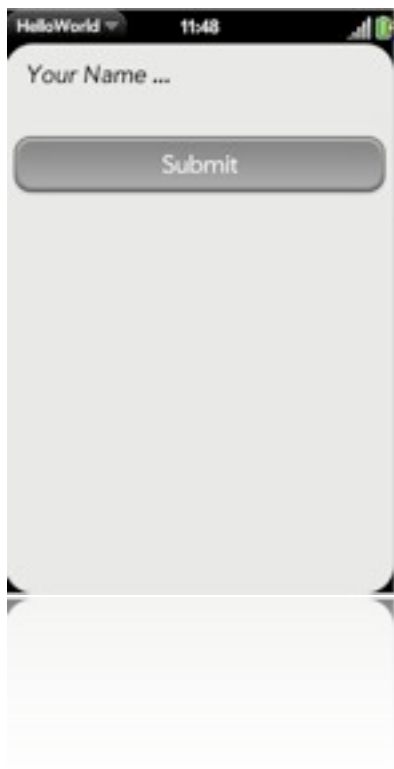
再想像一下這一次程式是如何的, 首先我如之前所說可以打我的名字, 再有一個Toggle Button 給人選擇給不給電腦記下我的名字, 之後有Submit, 更可以忘記我的名字. (所謂的記下, 只是寫入Cookie, 而非Database, Database 是後話)

### • 步驟 1:

先設定好Views, 記不記低Views 是什麼位置, 不記得就再提一提了:

HelloWorld>app>views>say-hello>say-hello-scene.html

先想像一下想砌的UI 是怎樣樣子的:



最後更新日期: 01/08/2009

我個人比較喜歡做簡單的東西 (看我學習HelloWorld 就明白了), 先做Forget Me 的按鍵. 想到如何做嗎? 和submit 相差不遠, 先在UI 上做一個forget 的DIV 欄位, 插在submit 的下面:

```
<!-- 這是新增的forget 鍵位置 -->
<div id="forget" x-mojo-element="Button">
</div>
```

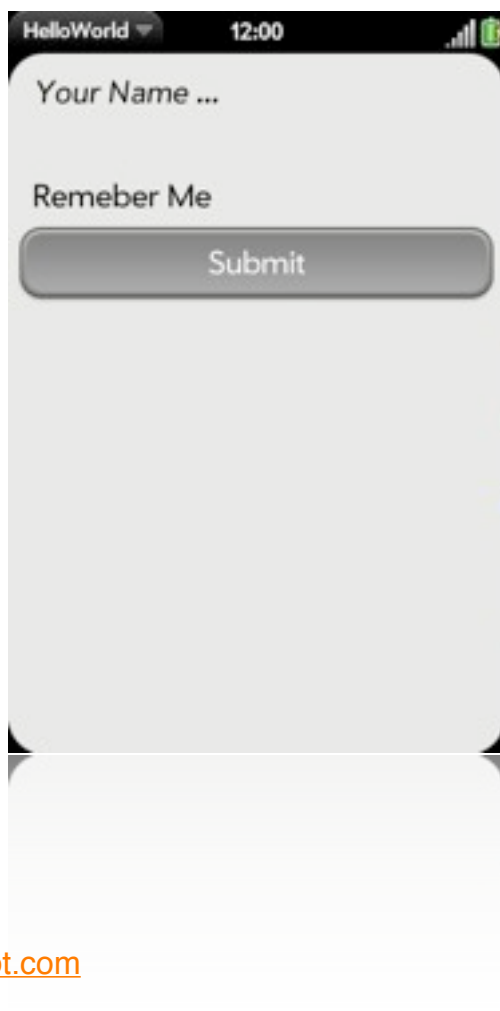
#### • 步驟 2:

就是做這個Toggle Button 和那個Remember Me 的Label 出來

```
<!--做一個palm-row-wrapper style 的Div 出來-->
<div class="palm-row-wrapper">
  <!--做一個Toggle Button 放的放置-->
  <div id="remember" x-mojo-element="ToggleButton">
  </div>
  <!--class="title left palm-text-wrapper" 就是之前所說的整好文字的工具
  而中間包住的Remember Me, 就是在ToggleButton 前面的文字
  -->
  <div class="title left palm-text-wrapper">
    Remember Me
  </div>
</div>
```

#### • 步驟 3:

如果現在行這個程式會看到這個樣子, 樣子不是理想中的那個, 因為未設定Mojo 的Widget, 還記得在那設定嗎? 不記得就要記住, 因為這個動作很常用: HelloWorld > app > assistants > say-hello-assistant.js 中的SayHelloAssistant.prototype.setup



先設定一下forget Button, 這個比較易, 和之前很像, 將這個Code 加在setup submit Widget 的後面便可以了:

```
/*forget Button*/  
this.controller.setupWidget("forget", this.attributes = {}, this.model = {  
    buttonLabel: "Forget Me"  
});
```



#### • 步驟 4:

做Toggle 的UI 之前, 我想做一個Cookies 和儲下使用者會不會記下Cookies 的選擇. 即使我要儲一個布林值(Boolean), 一個Cookies object. 不過在Javascript, 這些Data Type 的重要性不大, 我只要知要這兩個東西就行了. 在SayHelloAssistant() 的Constructor 上, 做兩個變數(variables), 分別是remember 和 cookie.

```
function SayHelloAssistant(){  
    this.remember; /*儲低使用者的選擇*/  
    this.cookie; /*儲低Cookies Object*/  
}
```

在Setup 時, 我們將我們的東西assign 到這兩個變數之中. 這句寫在setup 內的最頭頂

```
SayHelloAssistant.remember = true; //先將第一次出現的記名字的選擇設為True  
SayHelloAssistant.cookie = new Mojo.Model.Cookie("myCookies"); //透過Mojo 的Model 做一個Cookie,  
以"myCookies" 為id
```

• 步驟 5:

我要將Toggle Button 出來, 而做一個Toggle Button 和做Text Field 一樣簡單, 我將這放在 forget button 下面, 因為我比較愛同一類的UI 放在一群.

```
/**Toggle Button**/  
this.controller.setupWidget("remember", this.attributes = {  
    trueLabel: 'Yes', //是的Label 是"Yes"  
    falseLabel: 'No', //非的Label 是"No"  
}, this.model = {  
    value: SayHelloAssistant.remember, //用 remember 設定初設時的數值  
});
```



• 步驟 6:

做了基本動作, 就要做一些邏輯上的問題, 我們先處理如果沒有Cookies 記下使用者名字要如何處理先:

```
var displayName = ""; //新增一個變數, 為顯示使用者的名字  
  
if (SayHelloAssistant.cookie.get() != null) { /*如果在cookie 中拿到的不是null (即時使用者容許電腦記下名字)*/  
    displayName += SayHelloAssistant.cookie.get().yourName; /*將cookie 拿出來, 並將yourName 拿來 assign 到displayName 之上*/  
}
```

將displayName (要顯示的名字), 換到之前this.model 內的value, 即是變成以下這樣:



```
this.model = {  
    value: displayName, //初始時Textfield 的字  
    disabled: false //可以修改  
};
```

#### • 步驟 7:

將forget button assign 一個listener, 當有人tap 時, 行forget method, 而當remember widget (那個toggle button) 的變數改變時, 就行remember method

```
this.controller.listen("forget", Mojo.Event.tap, this.forget.bind(this));  
  
this.controller.listen("remember", Mojo.Event.propertyChange, this.remember.bind(this));
```

#### • 步驟 8:

之前原來我做少一樣東西, 就是沒有在cleanup 中清走Listener, 這個是一個不好的習慣, 因為這會用電, 又會令機拖慢, 所以現在補回, 日後才加回在第1天的步驟中, 我們在Cleanup 上將Listener 釋放出來.

```
Mojo.Event.stopListening(this.controller.get("submit"), Mojo.Event.tap, this.sayHello);  
Mojo.Event.stopListening(this.controller.get("forget"), Mojo.Event.tap, this.forget);  
Mojo.Event.stopListening(this.controller.get("remember"), Mojo.Event.propertyChange,  
this.remember);
```

#### • 步驟 9:

將sayHello 的Method 大改造:

```
//先做一個Geeting 的句子  
var geeting = "Hello " + this.controller.get("yourName").mojo.getValue() + "!";  
  
if (SayHelloAssistant.remember) { //如果使用者選擇記下名字  
  
    /*將cookies, 儲入一個JSON-  
    yourName, 內容則是 id yourName 的Value*/  
    SayHelloAssistant.cookie.put({  
        yourName: this.controller.get("yourName").mojo.getValue()  
    });  
  
    /*彈出一個errorDialog,  
    * 說出有關的Geeting 語句和說給使用者知下次會記下他的名字  
    */  
  
    Mojo.Controller.errorDialog(geeting + " I remember your name next time!");  
}  
else {  
  
    /*彈出一個errorDialog,  
    * 說出有關的Geeting 語句和說給使用者下次不會記下他的名字  
    */  
    Mojo.Controller.errorDialog(geeting + " Execute Me, What is your name?");  
  
}
```

- 步驟 10:

我們雖然寫好了 sayHello, 不過還有東西未完成, 就是使用者按 forget Me 的按鍵和改變是否記下名字的變更未做. 我們之前Listener 說了, 當我們的Toggle Button 有變動時, 便會觸發 remember 的Method, 現在我可以這樣做:

```
SayHelloAssistant.prototype.remember = function(event){  
    SayHelloAssistant.remember = event.value; //將觸發的event的value assign 到remember 變數之上  
}
```

- 步驟 11:

為forget Me 按鍵的tap 動作作出forget method 作回應.

```
SayHelloAssistant.prototype.forget = function(){  
    SayHelloAssistant.cookie.remove(); //將cookie 記錄移走  
    this.controller.get("yourName").mojo.setValue(""); //將Text Field 的名字轉成空白  
    Mojo.Controller.errorDialog("Oh No! What is your name?"); //彈出Alert Dialog 提使用者  
}
```

最後更新日期: 01/08/2009

讓我自己Demo 一次吧:

剛開是沒有名字的, 要輸入Your Name.... 之後輸入名字, “Ted WONG”, 再按 Submit 之後按OK > Fn + < (Windows 用家按 Home 鍵), 將卡飛走.



再開的話便會把Your Name 的位置自動填上 “Ted WONG”. 當然, 如果上一步將Remember Me 選成 No 的話, 你便不會看到自己的名字啦.



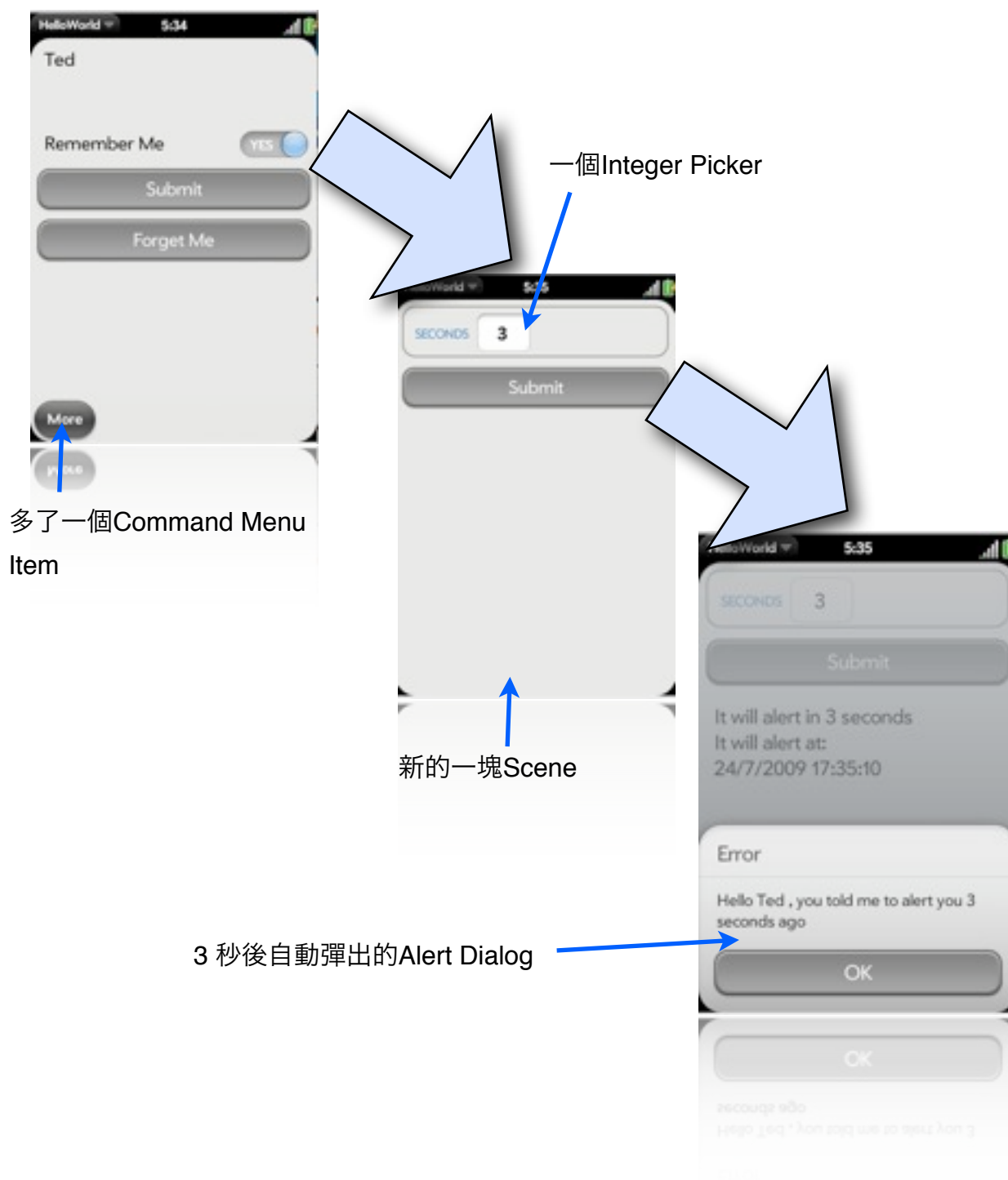
按Forget Me 就會將Cookies 清除, 下次開的時候便沒有填上名字了.



## 第 3 天: 預定時間的HelloWorld

到HelloWorld 開發的第三天了, 今天有點累, 希望可以在明天之前更新這份筆記. 今次本來想做一個Notification 的, 怎料這個Services 不是想像中的容易, 不過已似懂非懂, 明天再努力. 說回今天, 今天做的是做一個Command Menu 有一個Label 叫 “More”, 之前兩天我們做的都只是一塊Scene, 現在我們透過Command Menu, 由一塊Scene, 轉到另一塊Scene. 除非之外, 就是增加一個 Integer Picker, 給使用者選擇下次彈出Alert Dialog 的秒數. 到既定的秒數之後就會彈一個Alert Dialog.

今天做一個示範先:



## • 步驟 1

今次做一個Command Menu 先, Command Menu 和其他UIWidget 相似的地方就是都是用 controller 做setup 的, 而且這不用先在html 檔案上預定位置, 而且Command Menu 是要將其 Visible 轉成true 才可以看到的.

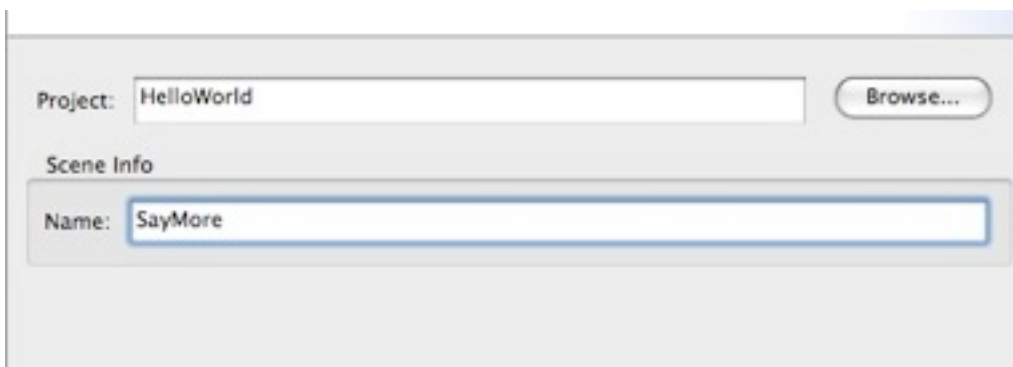
```
/*這是一個 Command Menu, 你可以做多過一個item, 現在只做一個,
 * label 是 "More", 按的時間會有一個叫sayMore 的event command 產生
 */
this.controller.setupWidget(Mojo.Menu.commandMenu, undefined, {
  items: [{
    label: "More",
    command: "sayMore"
  }]
});

//令Command Menu 變得Visible
this.controller.setMenuVisible(Mojo.Menu.commandMenu, true);
```

## • 步驟 2

我們為了讓這個Command Menu Item 可以呼叫其他 Scene. 所以我們需要做一個新的 Scene, 重覆第一天做的東西.

在 Project Explorer 中的HelloWorld Folder 右手鍵 > New > Mojo Scene  
輸入以下資料, 做一個新的Scene, 名為 SayMore



重溫一下, 我們多了say-more-assistant.js 在assistants 文件夾, 在views 內多了 say-more 的文件夾, 在這個文件夾中多了一個 say-more-scene.html 在sources.json 多了這段東西:

```
{
  "scenes": "say-more",
  "source": "app\\assistants\\say-more-assistant.js"
}
```

### • 步驟 3

既然我們做了一塊新的Scene, 那就砌這個Scene 的UI, 之後我們才處理command 跳轉新Scene 的動作.

這個UI 和之前的差不多, 只是多了一個Integer Picker 的預留位置, 如果不明白就看回之前做 x-mojo-element 的注解, 其次是多了一個div 用來扮作console, 顯示一些信息, 方便了解程式的運作.

```
<div class="palm-group unlabeled">
  <div class="palm-list">
    <!--這是Integer Picker 的位置-->
    <div id="secondPicker" x-mojo-element="IntegerPicker">
    </div>
  </div>
</div>
<div id="submit" x-mojo-element="Button">
</div>
<!--用來顯示文字用-->
<div id="console" class="palm-text-wrapper">
</div>
```

### • 步驟 4

今次設定UI 的Javascript 有點不同, 我們先做SayMoreAssistant 的Constructor , 我們需要從第一塊say-hello 那個Scene, 將使用者的名字轉到去say-more 的Scene, 所以我們做一個 username, 和一個alertSeconds 來記下會等待多久才彈出Alert Dialog. 我們假想到第一個 Scene 會給我們一個name, 所以將原先的

```
function SayMoreAssistant(){
}
```

換成這個

```
function SayMoreAssistant(name){
  this.userName = name;
  this.alertSeconds;
}
```

### • 步驟 5

至於有關 Setup 就和之前的差不多, 我也不多解釋:

```
//初始化Integer Picker
SayMoreAssistant.alertSeconds = 9;

this.controller.setupWidget("secondPicker", this.attributes = {
  label: "seconds",
  min: 3, //範圍由 3 - 60
  max: 60,
}, this.model = {
  value: SayMoreAssistant.alertSeconds,
});

this.controller.setupWidget("submit", this.attributes = {}, this.model = {
  buttonLabel: "submit"
});
```

```
//controller 的設定
this.controller.listen("secondPicker", Mojo.Event.propertyChange, this.setAlert.bind(this));
this.controller.listen("submit", Mojo.Event.tap, this.checkAlert.bind(this));
```

在clearup的method上,也和之前十分相似:

```
Mojo.Event.stopListening(this.controller.get("secondPicker"), Mojo.Event.propertyChange,
this.setAlert.bind(this));
Mojo.Event.stopListening(this.controller.get("submit"), Mojo.Event.tap,
this.checkAlert.bind(this));
```

## • 步驟 6

現在要設置的是將command menu item 按鍵時作出的回應. Command Menu Item 和其他 Widget 不同的是不用做Listener, 不過SayHelloAssistant 便多了一個Method 去處理這些 Command:

```
SayHelloAssistant.prototype.handleCommand = function(event)
```

至於內容則是測到有sayMore的event command 便轉到say-more的scene

```
SayHelloAssistant.prototype.handleCommand = function(event){
  if (event.type == Mojo.Event.command) { //event type 是Mojo.Event.command 的話
    switch (event.command) {
      case "sayMore": //command 是 sayMore 的話
        //將程式的畫面轉到say-more scene, 並而將使用者的名稱pass 到say-more 的scene
        Mojo.Controller.stageController.pushScene("say-more",
        this.controller.get("yourName").mojo.getValue());
        break;
    }
  }
}
```

讓我簡單地重覆一次整個過程:

Command Menu 被按 > More  
Item 會產生一個sayMore的Mojo

Event Command >  
SayHelloAssistant 的

handleCommand 會處理

Command 發出的Event > 經過一堆if, switch case > Mojo Controller 將 say-more scene 呼叫出來, 並同時傳送使用者名字的value (name) 到say-more scene > say-more scene 被載入, 將say-hello pass 來的name, 變成自己的userName

參照一下步驟8, params 會被傳到這個Method

```
"id": "com.blogspot.pre-dict",
"params": {"message": " " + "Hello " + this.userName + this.userName
+
" , you told me to alert you " + SayMoreAssistant.alertSeconds +
" seconds before" + ""}
```

## • 步驟 7

現在我們處理好如何轉Scene了, 你可以行來試試. 現在要處理的是Integer Picker, 和約定時間自動彈出Alert的部份了. 我們在步驟5已經做了controller, 當中的Integer Picker 會叫setAlert的method, 而按submit時會叫checkAlert method, 先做一個簡單的setAlert.

```
//當Integer Picker 的數值改變時, 呼叫這個method,
```

最後更新日期: 01/08/2009

```
//並將有關的數值assign 到alertSeconds 上  
SayMoreAssistant.prototype.setAlert = function(event){  
    SayMoreAssistant.alertSeconds = event.value;  
}
```



## • 步驟 8

現在要設定好要做的動作, 這個就是將Alert Dialog 彈出, 並說出"Hello *yourname* , you told me to alert you 10 seconds before":

```
var params = {
    "key": "alertKey", //這個是為了撞不同的Service 而設的
    "uri": "palm://com.palm.applicationManager/launch", //Launch Application
    "params": '{"id":"com.blogspot.pre-dict","params":{"message":"' + "Hello " +
this.userName +
    " , you told me to alert you " +
    SayMoreAssistant.alertSeconds +
    " seconds before" +
    "'}}', //com.blogspot.pre-dict 會Launch, 並傳送params
}
```

另外就是將有關的params 等設定好.

```
var d = new Date((new Date()).getTime() + parseInt(SayMoreAssistant.alertSeconds) * 1000); //將Date
加入所需秒數
```

```
params["at"] = (d.getUTCMonth() + 1) + '/' + d.getUTCDate() + '/' + d.getUTCFullYear() +
" " +
d.getUTCHours() +
":" +
d.getUTCMinutes() +
":" +
d.getUTCSeconds(); //將整個時間, 加到params ["at"], 即是在預定時間便彈出程式,
//如果程式在運行, 便會在彈出一個Alert Dialog
```

將params 傳到入service :

```
//通知Palm Service 在既定時間作出既定動作, 如果Service 成功便呼叫 handleOKResponse,
//失敗便呼叫 handleErrResponse
this.controller.serviceRequest('palm://com.palm.power/timeout', {
    method: "set",
    parameters: params,
    onSuccess: this.handleOKResponse.bind(this),
    onFailure: this.handleErrResponse.bind(this)
});

//在console DIV 顯示出以下信息
$("console").innerHTML = "It will alert in " + SayMoreAssistant.alertSeconds + " seconds
<br/>";
```

## • 步驟 9

至於這個成功與否的method, 都是單純改變一定console DIV 的內容已而

```
SayMoreAssistant.prototype.handleOKResponse = function(response){
    var d = new Date((new Date()).getTime() + parseInt(SayMoreAssistant.alertSeconds) * 1000);

    $('console').innerHTML += ("It will alert at: <br>" + d.getDate() + '/' + (d.getMonth() + 1) +
    '/' + d.getFullYear() +
    " " +
    d.getHours() +
    ":" +
    d.getMinutes() +
    ":" +
    d.getSeconds());
}

// This function handles the failed case
SayMoreAssistant.prototype.handleErrorResponse = function(response){
    $('console').innerHTML +=("Error service response: <br><br>" + Object.toJSON(response));
}
```

## • 步驟 10

我們要做一個App Assistant 來處理剛才寫好的code, 當時間到時, 這個params 便會傳到程式上. 那先製造一個app-assistant.js 在assistant.js 的folder 上, 再者, 在Sources 加入

```
{"source": "app\\assistants\\app-assistant.js"},
```

在app-assistant.js 之內, 我們要控制當程式在運行中我們如何處理, 所以我們要這樣寫

```
function AppAssistant(appController){
}

AppAssistant.prototype.handleLaunch =

function(params){

};
```

## • 步驟 11

最後便是將params 的message 透過Alert Dialog 彈出來便OK

```
AppAssistant.prototype.handleLaunch = function(params){
    if (params != null) {

        Mojo.Controller.errorDialog(params.message);

    }
};
```

## 第 4 天: HelloWorld 日誌

讓我先解釋一下“日誌”這一個字給香港的朋友聽, 日誌即時Log. 明白了吧? 我們之前為我們的 HelloWorld 做了不少東西, 今次要做的是記錄使用者輸入的文字. 今天我會集中學習幾樣東西, JSON , Javascript 的一些小技巧(真是一些, 不多), 及Depot 來儲存東西.

JSON 是一個將Object 變成一串String 的方法, 在 WebOS 很多時間都會用到, 所以我今天會花多一點時間學習 JSON, 不過其實之前幾天都用過, 而且很易明白, 所以所說的都不多. 重點是學習如何使用, 背後的運作, 便自己慢慢研究.

Javascript 方面, 會看看Eval method 和 Array 的使用, 如果是 Javascript 老手, 不要挑戰我呢, 我只是交流一下罷了, 真是一些簡單的用法罷了.

最後是有關WebOS 的儲存, 這個不是早就做了嗎? 我們不是用了Cookies 的形式處理嗎? 今次我們用的是HTML5 的Database, Depot, 有別不同的.

這個程式十分簡單, 比之前少UI 的東西(明天再學多點UI 的東西), 只是多了兩個Div 來看文字的輸出罷了. 如下圖:

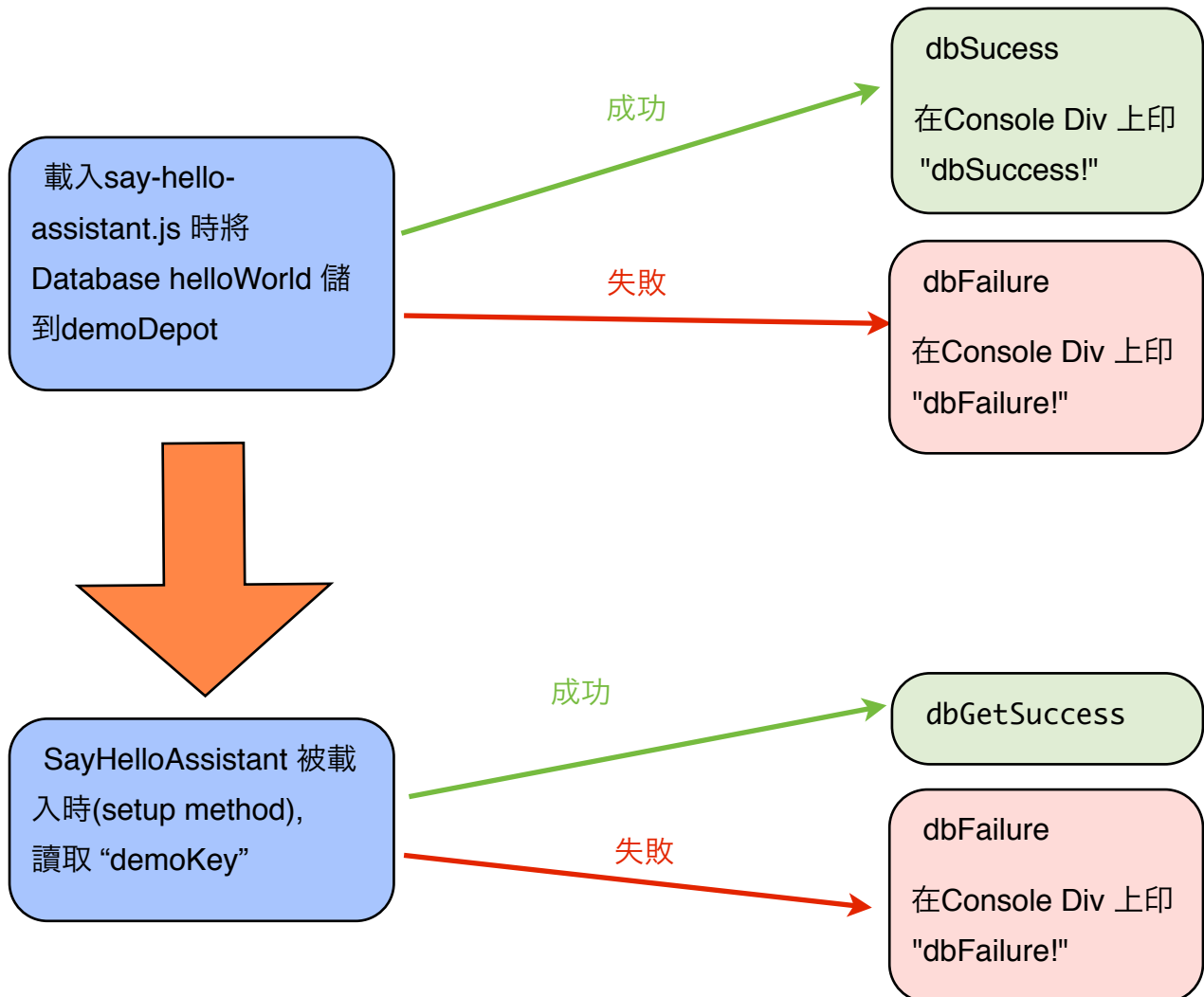


## • 步驟 1

一如以往, 先做 UI 的設定. 在say-hello-scene.html 中加入:

```
<!--用來顯示使用者輸入的log-->
<div id="log" class="palm-text-wrapper">
</div>
<!--用來顯示Depot Database 現在的情況-->
<div id="console" class="palm-text-wrapper">
</div>
```

在步驟 2 之前先解釋一下我想像中的程式是如何運作的:



而我們儲下的人名, 會以一個Javascript Array 作儲存. 利用JSON 將這變成一條String, 再用 eval method 來轉回一個Javascript Array.

在dbGetSuccess 的method 入面, 如果database 沒有東西的話, 在log variable assign 一個新的 Javascript Array, 並以JSON 的形式儲存. 反之, 將 database 的資料拿出來, 並儲到log variable, 之後將有關的資料轉會Array, 並一個一個userName 的log 印回來. 總之記著, 用的就是Array, 儲的是JSON.

當使用者按一下sayHello 的時間, log variable 會被讀取, 並變成Array 使用, 做一條新的 Array 再儲回Database 中. 先看看例子:

## • 步驟 2

要設定一個Depot database 來紀錄 logging. 先解釋一下, 我們之前用的Cookies 儲存方法的不足, 就是這個Cookies 是不可以用過 4KB 的Data. 而且Cookies 只是作臨時之用, 不足以做 logging, logging 應該是較長時間的, 所以我們用Depot database 儲起. Depot database 是 HTML5 新的東西, 可以給網頁開發者容易地作大量儲存. 當然WebOS 亦支援SQLite, 不過這不是今天討論的. 在Constructor (SayHelloAssistant())中, 加入以下的源碼:

```
//做一個log 檔來存下log 的內容(不用經常進取Database)
this.log;

/*開一個Database 名叫helloWorld
 * 用HTML5 database version 1
 * replace: false 的意思是如果有這個 Database 便不用覆蓋
 */
var options = {
  name: "helloWorld", //Name used for the HTML5 database name. (required)
  version: 1, //Version number used for the HTML5 database. (optional, defaults to
1)
  replace: false // open an existing depot
};

// 做一個Depot database, 成功的話行dbSuccess method,
// 反之, 行dbFailure. 並將這個depot database 放到 demoDepot 上.
this.demoDepot = new Mojo.Dpot(options, this.dbSuccess, this.dbFailure);
```

大家可能覺得好奇, 為什麼今次的Database 設置會早早在Constructor 設定, 而非在 SayHelloAssistant.prototype.setup() 中設置尼? 這個我不能解釋給你知(我會去找原因, 不過不是今天), 只可以說是一定要那裡做, 你可以試試在setup 中設置, 這一定會失敗! 不相信就自己試試.

### • 步驟 3

今次在setup method 中做的東西比較少, 只是在Database 取出有關數值:

```
//在demoDepot 用demoKey 取出數值, 成功則去dbGetSuccess, 反之去dbFailure
this.demoDepot.simpleGet("demoKey", this.dbGetSuccess, this.dbFailure);
```

### • 步驟 4

我們之前承諾了要用dbSuccess, 來表示成功連上database, 並在console 中顯示有關db 的狀況, 而dbFailure 亦十分相似. 最後便是成功在database 取出資料之後的邏輯, 便在dbGetSuccess 中實現.

```
SayHelloAssistant.prototype.dbSuccess = function(){
    //在Console Div 上印 "dbSuccess!"
    $("console").update("dbSuccess!");
}

SayHelloAssistant.prototype.dbFailure = function(transaction, result){
    //在Console Div 上印 "dbFailure!"
    $("console").update = ("dbFailure!");
}

SayHelloAssistant.prototype.dbGetSuccess = function(response){

    var recordSize = Object.values(response).size(); //database 得到資料的大小

    $('log').innerHTML = ""; //請空logging

    if (recordSize == 0 || response == undefined) { //database 內沒有demoKey資料
        $('console').update("No such record in the database");
        SayHelloAssistant.log = Object.toJSON(new Array()); //將一條新的Array assign 在log
    }
    else {
        SayHelloAssistant.log = response.userName; //有資料就將這資料assign 在log

        //因為儲的是JSON, 所以要將這轉成 JS Array
        var tempArray2 = eval('(' + response.userName + ')');
        //將Array 資料印出來
        for (var i = 0; i < tempArray2.length; i++) {
            $('log').innerHTML += (tempArray2[i] + "<br/>");
        }
    }
}
```

## • 步驟 5

至於最後, 我們要做的是處理使用者按Submit 時, sayHello 的問題. sayHello 是要將有關的資料轉成一條JS Array, 由於我們程式在setup 時, 已將database 將料取出, 並Assign 到 SayHelloAssistant 中的log 了, 所以我們不用再經database, 直接進調用SayHelloAssistant 中的log 便行了. 再者, 將這個資料變成Array, 再加資料成為一條新Array, 再assign 回 SayHelloAssistant, 再把這個Array 變回JSON 再放回database 便可以了. 有點複雜? 看例子, 在sayHello 中加入以下代碼:

```
var tempArray = eval("(" + SayHelloAssistant.log + ")"); //將log 變回JS Array Object
tempArray[tempArray.length] = this.userName; //Array 加入使用者輸入的名字

this.demoDepot.simpleAdd("demoKey", this.data = {
    "userName": Object.toJSON(tempArray)
}, this.dbSuccess, this.dbFailure); //將新的Array 寫入Database

this.demoDepot.simpleGet("demoKey", this.dbGetSuccess, this.dbFailure); //重新讀回database 的資料
```

沒有步驟 6 了, 這一天做的東西比較少. 不過就是學一學幾樣東西, 就是JSON 和 JavaScript 的Array 概念, 如果你明白上面的代碼, 你大可不用理會了. 首先要學的是JSON:

在WebOS 中, 發現很多這類的代碼:

```
var something = {
    string: "I am Ted",
    number: 1,
    boolean: true
};
```

這就是所謂的JSON, WebOS 很多時就是用這個方法來傳送設置, 試想想一個問題, 你在編程時, 有很多時間會用到Object, 不過Object 是一個概念性的東西. 它存在電腦記憶體上, 但不能寫入Database, 你可以將他不同的資料儲入Database 的field.

上述的例子, 我們可以將資料儲到一個Database, 內有3個field, 儲不同的datatype. 不過有時我們很懶, 不想這樣做, 而且進取 Database 再將Database 的資料轉回Object 都很廢時. 有時我們只想儲一條String 來代替一個Object. (當然, 我的log 可以用Database, 每次加一個資料比較適合, 不過這畢竟是一個學習的筆記, 給我這樣試試吧.) 用String 來紀錄 Object, 我們便會用 JSON 的方法儲下. 其實你看到沒有什麼特別, 只是一個field, 配一個data. 只是你會得到一條大家都看得懂的String 罷了. 雖然簡單, 不過你一定要明白.

其次是如何將Object 變成 JSON, JSON 如何變回 Object. 因為 WebOS 是預載了 prototype.js , 所以這變得很容易. (當中亦有很多好用的東西, 如 \$("something"), 是 document.getElementById("something"); 的偷懶版.) 我們只要用:

Object.toJSON(obj)

便可以將 object 變成一條String 的形式表達了.

<http://www.pre-dict.blogspot.com>

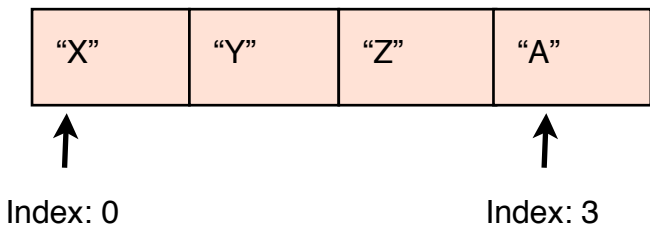
最後更新日期: 01/08/2009

當我們有 JSON, 我們便想將 JSON 變回 Object. 這個也很簡單:

```
eval('(' + jsonString + ')');
```

JavaScript 的一個eval method, 可以將JSON 變回 JavaScript Object, 當然, 要加入 “(“ 和 “)”. 否則這只會當String 來看待, 很強了吧?

最後一個要懂的是 JavaScript 的 Array, 有經驗的程式員一定知道, 什麼是 Array Index Out of Bound. 不懂的我介紹一下:



這是一個length 為4 的Array, 數一數, 有4格. 而電腦認的是Index, 由零開始, 最大的Index 是 3. 同樣道理, 你會知道最後的那格, 一定是 `array[array.length-1]`

如果在 Java 中你試試呼叫Index 4 的Array, 就一定會出現 Runtime Error. 不過在Javascript, 你一樣可以叫Index 4 那一格, 存回的會是undefined. 不過沒有Array Index Out of Bound, 你可以在 JavaScript 中叫 `array[array.length]` 來assign 一個新的數值進Array, 就像在後面加東西似的. 如之前的例子:

```
tempArray[tempArray.length] = this.userName; //Array 加入使用者輸入的名字
```

很方便了吧？不過千萬不要在其他語言用這個語法，你會後悔。

還有一點給初學者, Java 和 JavaScript 是兩樣東西. 記得在Palm 的developer forum 中, 有一句很好的:

"Java is to JavaScript as Car is to Carpet"



## 第 5 天: HelloWorld 日誌 +

今天之後, 基本上會開一個新的主題, 有關WebOS 和Internet 的, 所以第 5 天是最後一天基本的UIWidget 的應用. 這一天和昨天做的差不多, 亦是用回那條Array 來做東西, 所加的東西不多, 不過今次要學的是WebOS 可以用template 來設定UI Element 的樣子.

先看看有關的圖片:



解釋一下先, 這個是前幾天做的More-Assistant, 今次我們做個 List 給More Assistant, 而且今次顯示Log 和先前的不同, 這是一個可以Filter 的List, 而這個List 和 Say-Hello 的內容 log 是一樣的. 不過這個List 的顯示是我用一個Template 形式的html 設置的, 而且這是可以搜尋的, 只要我打一個e 字, 便可以找到只有e 字的Name. (這天我是參考很多Palm 的example) 好, 立即開始!

### • 步驟 1

首先要解決一個問題, 就是say-hello-assistant 之前只是pass一個數值, userName, 不過今次就不同了, 今次要傳多一樣東西. 沒錯, 就是SayHelloAssistant 的log Array, 你可以pass JSON 之後在SayMoreAssistant 將他變回Array, 或是直接pass 一個Array 都沒有問題. 我就採用後者, 所以在handleCommand 中, 改成這句:

```
//第五天: 連Array 也pass 到say-more scene
    Mojo.Controller.stageController.pushScene("say-more",
this.controller.get("yourName").mojo.getValue(), eval('(' + SayHelloAssistant.log+ ')'));
```

同樣的道理, 在say-more scene 的Constructor 改要修改一下, 成為以下的樣子:

```
function SayMoreAssistant(name, array){
  this.userName = name;
  this.alertSeconds;
  this.userArray = array;
}
```

再來一個簡單的UI 設置 (前幾天我可能分開幾個步驟, 今天寫在一個步驟, 希望大家習慣):

```
<!--Filter List 的位置-->
<div x-mojo-element="FilterList" id="filterLogList"></div>
```

## • 步驟 2

設置這個 FilterList 似是一個很簡單, 同樣pass 一堆JSON 進去便選了?! 錯! 這個是今天的重點, 這個設置真是整天最難的一個部份之一.

先做兩行先, 一行是處理Data, 第二行是Model 設disables 為false.

```
this.setupData();

this.model = {
  disabled: false //false, 即時容許Filter
};
```

現在處理的是如何setupData:

```
SayMoreAssistant.prototype.setupData = function(){
  this.data = []; //空的数据
  for (var i = 0; i < this.userArray.length; i++) {
    this.data[i] = { //將userArray 的資料存到data 的array
      "name": this.userArray[i], //將userArray 的內容變成name
      "number": "" + i //number 就是指第幾個
    }
  }
}
```

## • 步驟 3

這是如何setupWidget, 沒錯, 這需要一個步驟, 証明其難道. 這不和之前的 Widget 那麼容易 setup, 來, 我們看看怎樣設置.

```
this.controller.setupWidget('filterLogList', this.attributes = {
  itemTemplate: 'say-more/entry', //用say-more/entry.html 為template
  swipeToDelete: false, //可以被delete
  reorderable: false,
  filterFunction: this.list.bind(this), //用list method 來filter
  formatters: {
    name: this.formatName.bind(this), //用formatName 來設定Name
    number: this.formatNumber.bind(this) //用formatNumber 來設定Name
  },
  delay: 0, //在filter string 時作出0秒延遲
  disabledProperty: 'disabled'
}, this.model);
```

我們按次序解釋, itemTemplate 設定用 “say-more/entry”, 所以在HelloWorld/views/say-more/ 中要加入一個entry.html, 這個是UI 的設置, 我們這樣做:

```
<div class="palm-text-wrapper">
Number: #{numberFormatted}
</br>
Name: #{nameFormatted}
</br>
</br>
</div>
```

這正正是我們要用的template, 你可以看到 #{numberFormatted} 和 #{nameFormatted}, 這將會是我們的數據. 就是在setup Widget 中將format 好的數據放進去.

其次是list 的method 來作filter, 我們又看看這段怎樣寫: (這個method會在你打字時呼叫的)

```
SayMoreAssistant.prototype.list = function(filterString, listWidget, offset, count){
    var subset = [];
    var totalSubsetSize = 0;

    var i = 0;
    //一直loop, 直到數到data 中最後的一個element
    while (i < this.data.length) {
        //將name 變成UpperCase, 和filterString 變成UpperCase
        //即時不理會大細階, include 是指filterString 中有沒有name
        //這即時部份搜尋到都為有效
        if (this.data[i].name.toUpperCase().include(filterString.toUpperCase()) ||
            this.data[i].number.include(filterString)) {
            if (subset.length < count && totalSubsetSize >= offset) {
                //將資料push 進去
                subset.push(this.data[i]);
            }
            totalSubsetSize++;
        }
        i++;
    }

    //更新listWidget
    listWidget.mojo.noticeUpdatedItems(offset, subset);

    //如果filter 後的list 的長度和數量不一樣, 就要更新
    if (this.filter !== filterString) {
        listWidget.mojo.setLength(totalSubsetSize);
        listWidget.mojo.setCount(totalSubsetSize);
    }
    this.filter = filterString;
}
```

至於formatNumber 和 formatName 就易了解很多:

```
SayMoreAssistant.prototype.formatNumber = function(n, model){
    //將數字return 出去
    return n;
}

SayMoreAssistant.prototype.formatName = function(n, model){
    //Capitalize, 會將第一個英文字變成大階
    return String(n).capitalize();
}
```

#### • 步驟 4

接下來的便很簡單了, 就是做event handling, 在setup 中, 為 filterLogList 設置如下:

```
this.tapped = this.tapped.bindAsEventListener(this);
this.gotFilter = this.gotFilter.bind(this);
//tap 的話, 行以tapped method
Mojo.Event.listen(this.controller.get('filterLogList'), Mojo.Event.listTap, this.tapped);
//做filtering 時, 行got Filter method
Mojo.Event.listen(this.controller.get('filterLogList'), Mojo.Event.filter, this.gotFilter,
true);
```

之後在cleanup() 時做以下動作:

```
Mojo.Event.stopListening(this.controller.get('filterLogList'), Mojo.Event.listTap, this.tapped);
Mojo.Event.stopListening(this.controller.get('filterLogList'), Mojo.Event.filter,
this.gotFilter, true);
```

在say-more-assistant 加回 tapped 和 gotFilter method:

```
SayMoreAssistant.prototype.gotFilter = function(event){
    console.log("GOT FILTER EVENT IN CLIENT, str=" + event.filterString);
}

SayMoreAssistant.prototype.tapped = function(event){
    console.info("TAPPED ELEMENT " + event.item.name);
}
```

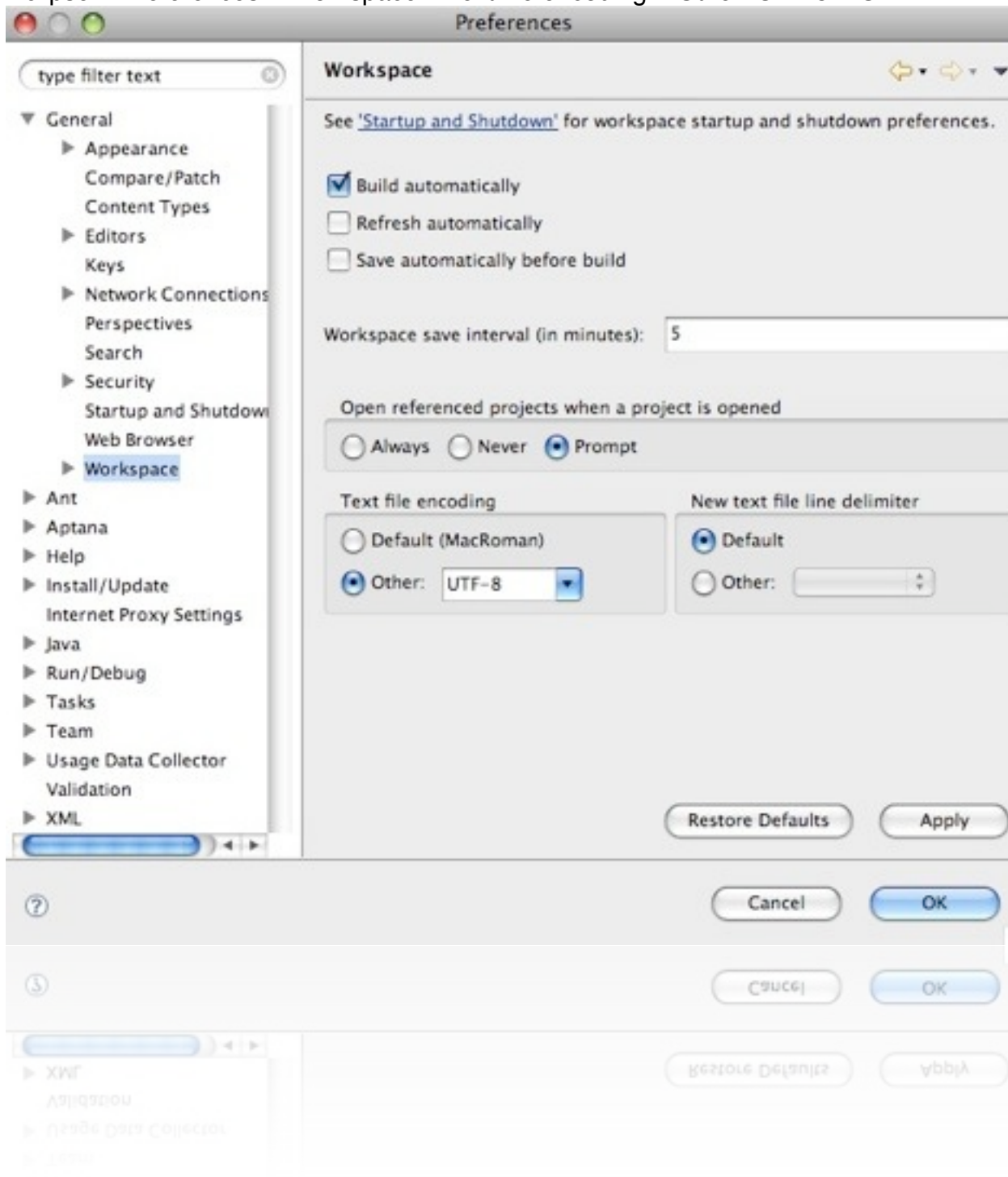
好了, 第 5 天的課程就是這樣. 明天就開始做些關於網絡的東西.

# 備忘

## Eclipse 內打中文

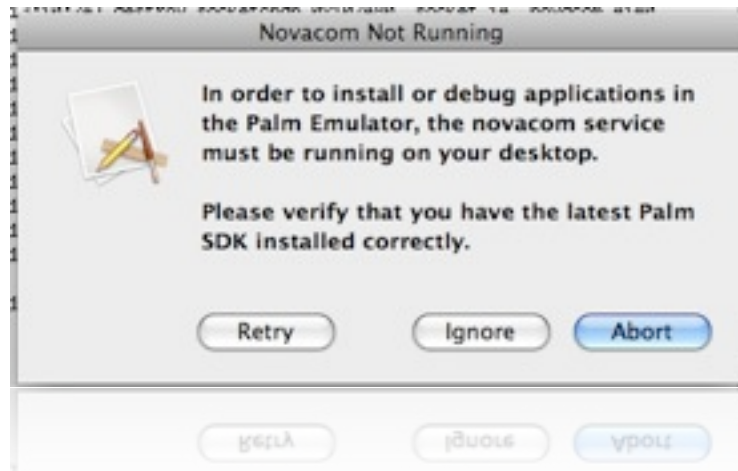
Eclipse 最初的設定是不容許在coding 當中打中文的, 如果要在Source Code 內打中文, 可以有以下做法:

Eclipse > Preferences > Workspace > Text file encoding > Other: UTF-8 > OK



# 除錯指南

## Novacom Not Running



觸發原因:

這個錯誤是更新了WebOS 1.1 和安下新SDK 之後出現的, 當使用Palm Emulator 便會出現描述:

不能安裝程式到Palm Emulator, 只是單純運行模擬器, Debug 不能, 重新安裝都不能解決.

解決方法: (29/7/2009 最新更新)

在Terminal 中打以下Command:

```
$ sudo chmod 644 /Library/LaunchDaemons/com.palm.novacomd  
$ sudo /opt/nova/bin/post-install.sh
```

重新啟動你的電腦, 便行了.

參考:

<http://www.ozmox.com/2009/07/25/webos-on-mac-os-x/>

# 更正內容

這個位置是我在不停更新時, 發現的錯誤, 不過因為未有時間更新先前的Source Code, 所以在這裡寫下, 待完成整個筆記之後才改正.

## Listener 正確的設置方法

在Palm 的Developer 論壇中幾經交流, 發現正確的Listener 設置方法如下:

```
this.eventHandler = this.sayHello.bind(this);  
this.controller.listen("submit", Mojo.Event.tap, this.eventHandler);  
  
this.controller.stopListening(this.controller.get("submit"), this.eventHandler);
```

## 處理彈出的Alert Dialog Box

參考:

<http://developer.palm.com/distribution/viewtopic.php?f=9&t=555>

# 有用連結

## 官方文檔 (SDK 相關)

Install Palm Mojo SDK: (OS X)

[http://developer.palm.com/index.php?option=com\\_content&view=article&id=1545](http://developer.palm.com/index.php?option=com_content&view=article&id=1545)

Developer Guide: (概念, 基礎知識)

[http://developer.palm.com/index.php?option=com\\_content&view=article&id=1645](http://developer.palm.com/index.php?option=com_content&view=article&id=1645)

Command Line Tools:

[http://developer.palm.com/index.php?option=com\\_content&view=article&id=1552](http://developer.palm.com/index.php?option=com_content&view=article&id=1552)

Palm Widget Document:

<http://developer.palm.com/palm-sdk/jsdoc/symbols/Mojo.Widget.html>

Palm 官方CSS:

[http://developer.palm.com/index.php?option=com\\_content&view=article&id=1630](http://developer.palm.com/index.php?option=com_content&view=article&id=1630)

## SDK 相關 (非官方資料)

Prototype API:

<http://www.prototypejs.org/api>

## 討論區收藏

Listener: (詳細閱讀)

<http://developer.palm.com/distribution/viewtopic.php?f=10&t=491&start=0>

## 基礎知識

JavaScript Array:

[http://www.hunlock.com/blogs/Mastering\\_Javascript\\_Arrays](http://www.hunlock.com/blogs/Mastering_Javascript_Arrays)

JSON in JavaScript:

<http://www.json.org/js.html>

## 除錯相關

Novcom Not Running:

<http://developer.palm.com/distribution/viewtopic.php?f=6&t=118>



# 版權聲明

Palm, Inc. Palm, Pre, Mojo, Synergy, and webOS are trademarks of Palm, Inc.