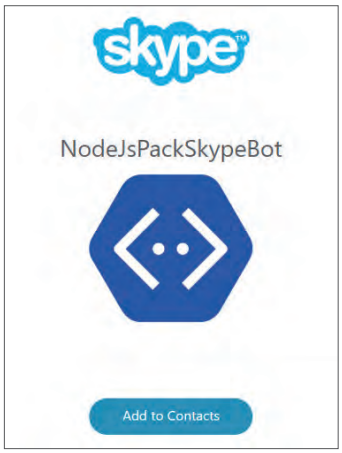
按下 Add to Skype 按鈕，以便把 Bot 加至 Skype 聯絡人清單，這個動作會打開一個新分頁或視窗，所看到的畫面如下：



請再按下 Add to Contacts 按鈕，這個動作會自動打開 Skype 軟體，並且把 Bot 加到我們的 Skype 聯絡人清單。

如果你的 Skype 與 Azure 使用不同帳號（或你已登出 Azure 帳號），這時你必須先登出再重新登入你的 Skype 帳號，才能順利把 Bot 加到聯絡人清單。

完成這些設定步驟之後，必須對程式碼做些修改，也就是 APP_ID 與 APP_SECRET 兩個變數，我們註冊 Bot 時所輸入的那組帳號密碼，就是這邊需要設定的內容。完成這項修改後，需要重新將 Bot 程式發佈到 Azure 才能測試。

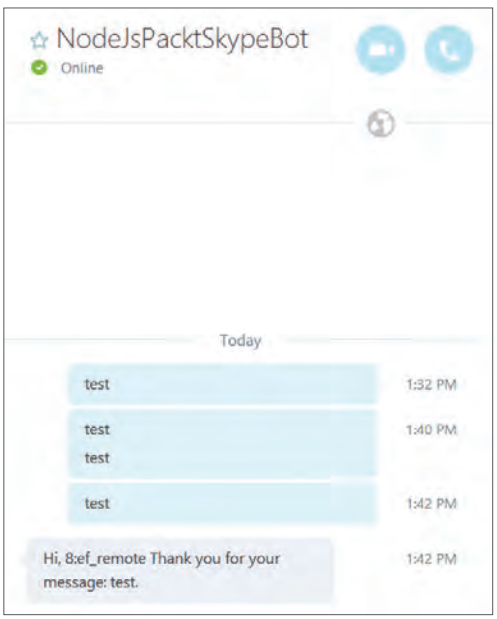
請參考以下對 Skype Bot 的 app.js 程式碼所作修改：

```
var skype = require('botbuilder');
var express = require('express');
var app = express();
var APP_ID = '<< Your Application Id >>';
var APP_SECRET = '<< Your Application Password >>';
var botService = new skype.ChatConnector({
  appId: APP_ID,
  appPassword: APP_SECRET
});
```

```
var bot = new skype.UniversalBot(botService);
app.post('/api/messages', botService.listen());
bot.dialog('/', function (session) {
  if (session.message.text.toLowerCase().indexOf('hi') >= 0){
    session.send('Hi ' + session.message.user.name + ' thank you for
your message: ' + session.message.text);
  } else {
    session.send('Sorry I don't understand you...');
  }
});
app.get('/', function (req, res) {
  res.send('SkypeBot listening...');
});
app.listen(process.env.port, function () {
  console.log('SkypeBot listening...');
});
```

修改過的網站程式發佈到 Azure 後，如果你已經成功把 Bot 加到 Skype 聯絡人清單，這時我們就可以送出訊息給 Bot，如同下圖中所見。

目前這個 Bot 會把收到的訊息回傳給我們，並附加友善的感謝詞。



現在，就來揭曉這背後的魔術，請留意 8:ef_remote 這串文字，它就是實際發送訊息給 Bot 的 Skype 使用者代碼。

程式碼中負責這段變魔術的就是 bot.dialog 事件。

從這個事件的名稱不難看出，它被觸發的時機點，就是當 Skype Bot 收到一個訊息時，會透過 HTTP POST 方式送來一個請求。請看以下的程式碼片段：

```
bot.dialog('/', function (session) {
  if (session.message.text.toLowerCase().indexOf('hi') >= 0){
    session.send('Hi ' + session.message.user.name + ' thank you for
your message: ' + session.message.text);
  } else {
    session.send('Sorry I don't understand you...');
  }
});
```

其中 session 物件包含 Skype 傳遞給 Bot 程式的資訊，它包含訊息內容與來源對象。請留意 session 物件中包含的屬性名稱，像是 message 與 text 等。

人力資源小助手

目前為止，我們已經建立基本的 Skype Bot 程式並部署至 Azure，它能回應收到的訊息，並且在回覆的訊息中附加感謝詞。

前一章我們曾略述一個 BotBrain 的作法，負責對特定訊息回覆方式的處理邏輯。

現在繼續為 Skype Bot 擴充程式功能，建立一個基本的人力資源（HR）小助手，它能回答特定問題，例如，檢查某個人還有多少休假或是請病假。

人資領域還包含很多其他議題，當然我們也可以加入更多程式邏輯，到自動化的人資小助手，但是本書的重點是說明通訊服務的自動化，為避免模糊焦點，我們的範例只侷限於簡單的休假與病假流程處理。

我們將利用 Azure 提供的 Table Storage（https://azure.microsoft.com/en-us/documentation/articles/storage-introduction），用於依據收到訊息的類型與使用者的要求，來定義 Bot 程式回答問題所需的資料。

Azure Table Storage 後端資料儲存

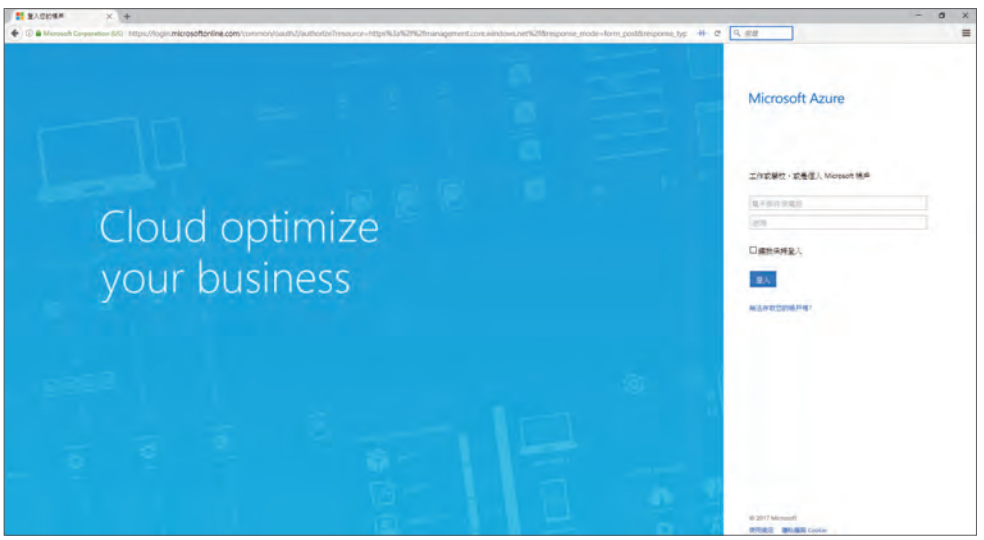
Table Storage 服務以表格形式保存資料，把每筆資料記錄視為實體（entity），表格欄位則代表實體的各種屬性（表格中的欄位）。

每筆實體資料有自己的一對鍵值（PartitionKey 與 RowKey）以便識別，它也有一個時間戳記欄位，以供 Table Storage 服務記錄每筆實體資料內容被更新的時間（時間戳記會由服務內部自動產生而且不能被修改）。

關於更多儲存體與 Table Storage 服務的運作方式，可以在 Azure 網站找到詳細說明文件（https://docs.microsoft.com/en-us/azure/storage/），裡面有許多寶貴的資源，仔細研讀可以更深入瞭解這些服務。

接下來我們會簡短說明如何讓 Azure Table Storage 開始運作。

想要使用 Microsoft Azure 的資料儲存服務，你需要先登入到 Azure Portal（http://portal.azure.com）。參考以下畫面：

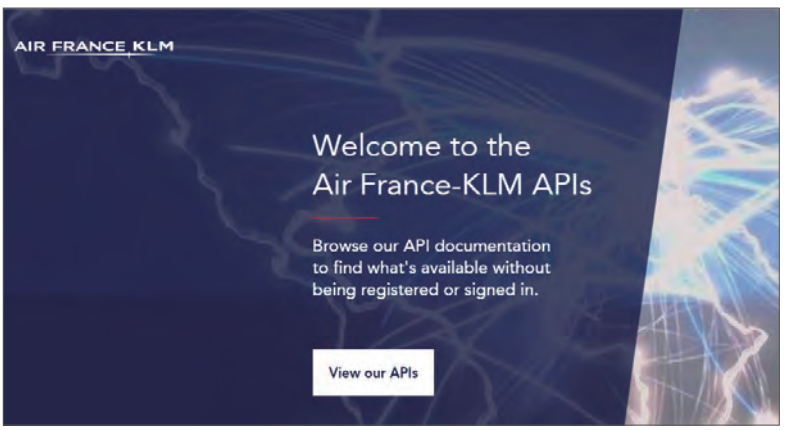


```
          console.log(error);
        }
        console.log(tweetReply.text);
      });
    });
    stream.on('error', function(error) {
      console.log(error);
    });
  });
```

程式修改完成後，接下來就可以加入發送航班資訊與狀態必要的邏輯處理。

航班資訊 API

為取得航班資訊，我們的程式需要串接航班 API 服務。我們在範例搭配的法荷航空 API，就是一個免費又好用的服務，可以在 <https://developer.airfranceklm.com/> 網站取得。



我們需要先註冊一組帳號。法荷航空 API 提供許多服務，例如：預約、訂購、航班優惠、航班狀態、地點、聯絡資訊、附加購物服務與報到手續等。（請參考 <https://developer.airfranceklm.com/Our_Apis>）

我們從第四章的 Slack Quote Bot 範例，可以見到 Slack 是一個很棒的團隊協作平台。在協作過程中，團隊成員可以獲得啟發靈感的箴言。在本章，我們將會看到一個更複雜的 Slack 使用案例，我們會使用 Howdy BotKit 建立一個叫做 DocMan 的 Bot，DocMan 能夠依照團隊成員的要求搜尋文件，並可以提供下載連結。

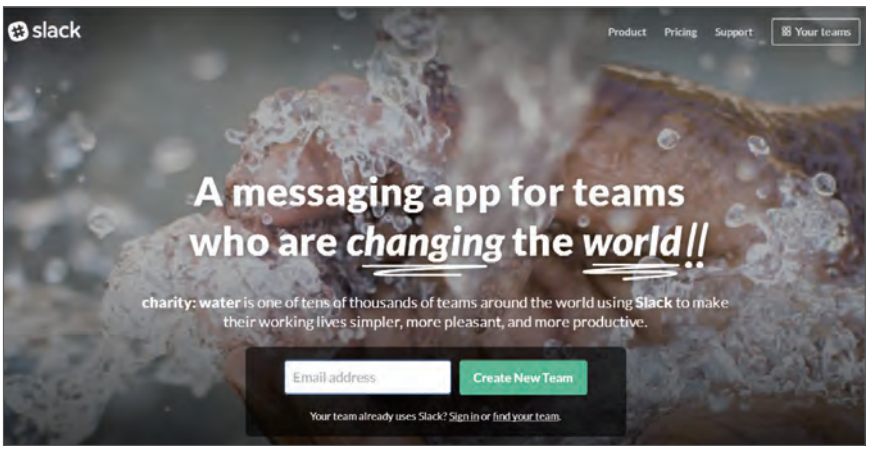
我們的 Slack Bot DocMan，會使用 MongoDB 來儲存資料，而文件與檔案會存放在 Amazon S3。關於 MongoDB 與 Amazon S3 的細節將會在本章稍後說明。

太棒了！我們現在開始使用 Slack 吧！

為你的團隊設定專屬 Slack 平台

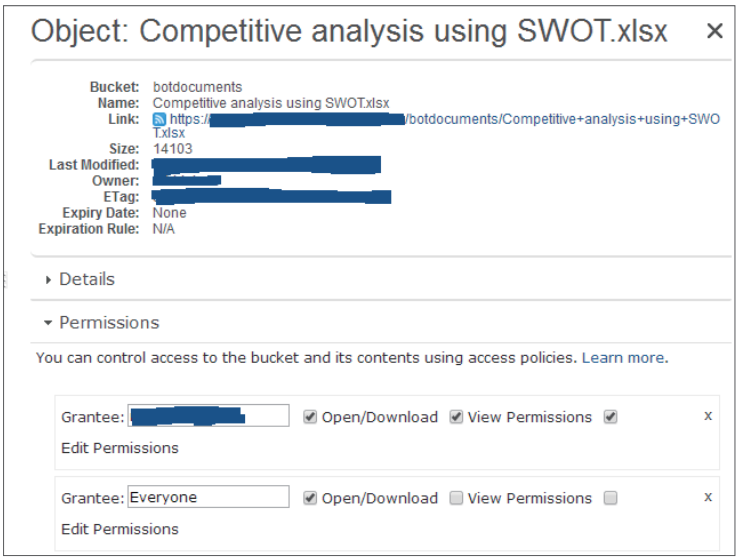
我們在此小節中，將開始設定團隊專屬的 Slack 平台。

請打開瀏覽器視窗，輸入網址：https://slack.com。你可以看到 Slack 網站的首頁畫面如下：



對於第一次接觸 Slack 的使用者，必須先建立個人的 Slack 帳號，然後建立該帳號關聯的團隊。已經擁有 Slack 帳號的使用者，可以直接點登入（Sign in）連結。我們先看看如何建立個人的帳號：

這動作會顯示所選文件的所有屬性，如下圖：



從屬性的資訊中，找到 Link 屬性，這就是我們所需的 URL。

用這個方法，就可以把所有文件都標示為 Public 並複製它們的 URL，然後更新 URL 到 MongoDB 資料庫中。

更新 MongoDB 的資料加入 Amazon S3 文件連結

再次執行 Mongo shell 存取 BotDB 資料庫，使用的指令如下：

```
Use BotDB db.ReferenceDocuments.update({
    title:"Competitive analysis using SWOT"
}, {
    $set: {
        url:"<YOUR AMAZON S3 URL FOR THIS DOCUMENT>"
    }
})
```

一旦更新成功，你會從 mongo shell 看到有幾筆記錄被更新。用相同的步驟，更新 Amazon S3 的公開連結到所有剩下的文件資料的 URL 欄位，如此便能完成 docman 所需的後端資料。

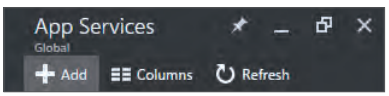
這個 Page Access Token 會在與 API 整合時使用。

Facebook App 需要透過 Webhooks 來接收使用者的訊息，在開始設定 Webhooks 之前，我們先建立一個用 Node.js 撰寫的 Facebook Messenger App。Bot 跟 Facebook 的 Webhooks 整合，需要使用 HTTPS 連線加密，可以使用 Microsoft Azure 來做到這一點。

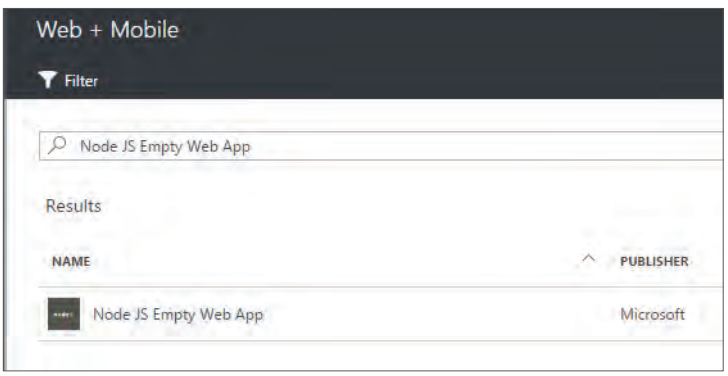
在 Azure 中建立 Bot 伺服器

如之前所提到的，需要一個支援 HTTPS 協定的 Bot 伺服器，這樣才能跟 Facebook 整合，所以，我們將在 Micsoroft Azure 中建立一個 Bot 伺服器。

前往 <https://portal.azure.com/> 並登入帳號，然後點選左側選單的 App Services 來建立一個基於 Node.js 的 Bot 伺服器。請參考下圖所示：

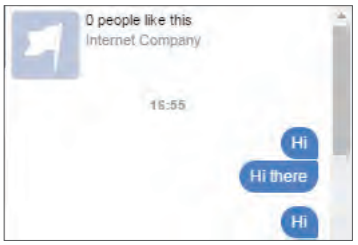


點擊 Add 連結將前往以下畫面，接著在 Web + Mobile 頁面的搜尋框填入 `Node JS Empty Web App` 然後按 enter 啟動搜尋，如下圖所示：



選擇 Node JS Empty Web App 後，點擊 Create 按鈕來建立一個空白 Node.js 樣板。下一步是需要輸入 App name 和 Subscription 等資訊，如下圖所示：

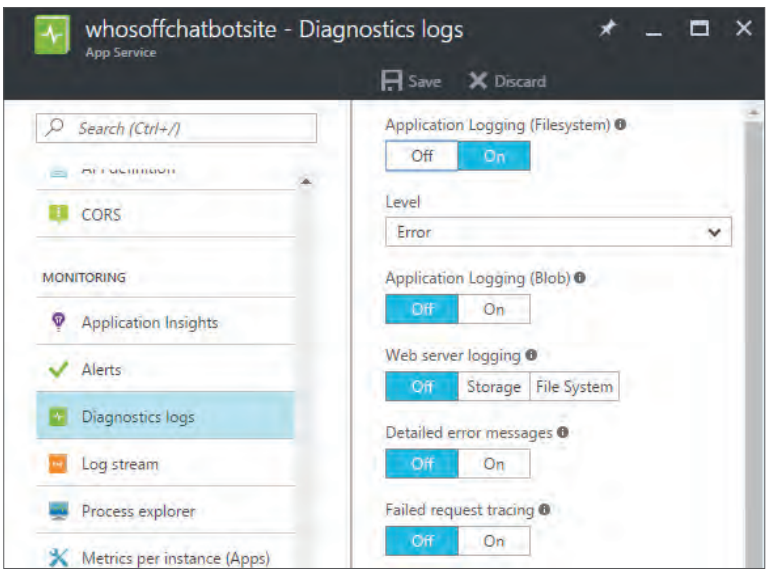
當你部署更新過後的程式碼，再次呼叫 Facebook 專頁，如下所示：



我們已經送出多則訊息給 Bot，但它並沒做任何事情。讓我們來看看 Azure 裡到底發生了什麼事，是否有任何應用程式層級的錯誤訊息。

在 Azure 環境裡的 Bot 故障排除

要解決 Bot 沒有任何回應的問題，首先，前往 Diagnostics logs 子選單，並且開啟 Application Logging，接著再前往 Log stream 子選單看看它是否有顯示應用程式的任何錯誤。



現在，對 Bot 發送訊息，並查看應用程式裡的 log，你應該會看到錯誤訊息以及發生錯誤的行號，如下圖所示：

DocumentDB 是什麼？

在第六章已經介紹過什麼是 NoSQL，Microsoft Azure 平台提供的 DocumentDB，也是一種 NoSQL 資料庫，它會以 JSON 格式儲存資料。

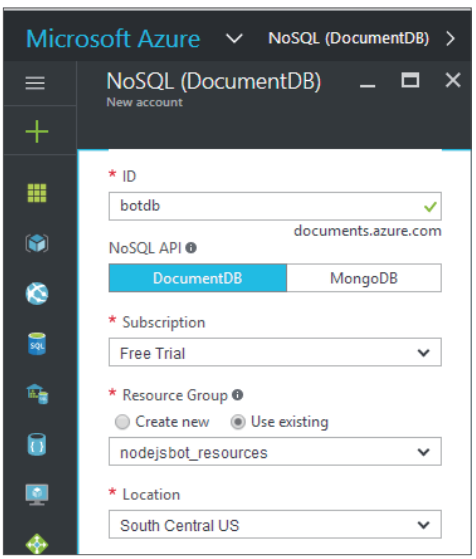
關於 DocumentDB 更多的細節，請參考 <https://azure.microsoft.com/en-in/services/documentdb/>。

為 Who's Off Bot 設定 DocumentDB

假設你已經有使用 Microsoft Azure 的服務，你可以根據以下的步驟，為你的 Bot 設定 DocumentDB 資料庫。

■ 為了 DocumentDB 建立帳號 ID

依據以下畫面在 Azure 建立一個新的 ID 叫做 `botdb`。選擇 DocumentDB 做為你的 NoSQL API，然後再選你預計要使用的訂閱方案（Subscription）與資源 (Resource)，在這裡你可以使用已存在的資源或新增一個新的資源。輸入完所有必要欄位後，按下方的 Create 按鈕為 DocumentDB 建立一個新的帳號。



基於預約的會議資料來顯示誰在什麼時段是忙碌的。

至此，已經完成了我們的 Bot，它在 Facebook Messenger 對話視窗上可以提供友善的操作介面來預約或查詢會議。我們還可以新增一個功能叫 My Schedule，這個部分就留給你當課後練習。

小結

透過 Facebook 可以建立增強團隊協作體驗的 Bot。我們可以透過聊天視窗傳送會議資訊給 Bot 程式，例如會議名稱、起始及結束日期，Bot 就會自動預約一場會議。

總結來說，我們已經學會了如何建立 Facebook 專頁以及 Facebook App。也透過 Node.js 來建立一個基本功能的 Bot，然後把 Bot 部署到 Microsoft Azure 上，因為 Facebook Messenger 的 Webhooks 整合需要使用 HTTPS 連線。然後我們訂閱了 Facebook 專頁的 Webhook，實作了 Webhook 的功能，於是 Bot 就能接收到使用者在 Facebook 專頁上發布的訊息。

最後是強化 Bot 顯示資訊的能力，像是透過 Facebook Messenger 介面，查詢誰在什麼時候休假。

我們也看到 Who's Off Bot 在預約會議時，聰明地檢查是否有會議時間衝突，並透過 Facebook Messenger 提供的顯示樣板元件，展示整個團隊的預約會議行程。

如果你想要開發更聰明的 Bot，可以進一步研究 <https://wit.ai/> 和 <https://api.ai/>。這些平台提供的服務，開發出更像真人的聊天機器人。

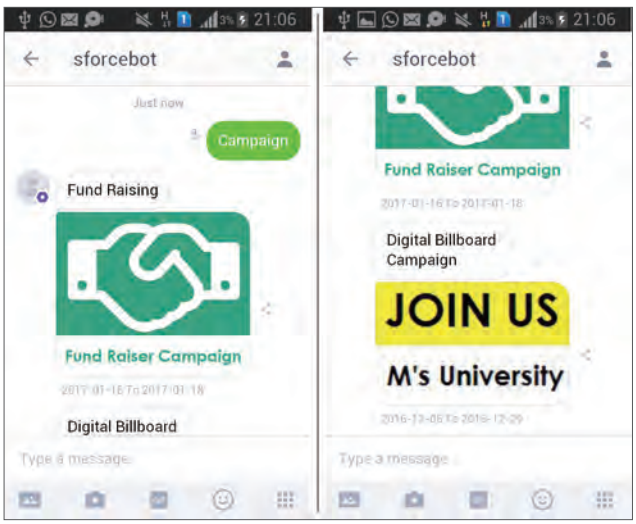
對於建立 Facebook Messenger Bot 的過程，希望本章能帶給你很棒的體驗！

下一章我們會繼續探索，如何使用 Node.js 開發與 IRC 服務整合的 Bot 程式，幫助開發者用它來追蹤專案的 Bug。

Sforcebot 的行銷活動管理

到目前為止，我們已經可以從 Kik 平台取得商機資訊，也可以取得 Salesforce 活動資料。假設某所大學正在使用 Salesforce 進行校園活動管理，這個大學想要透過 Bot 推廣活動，確保學生跟學校之間進行的各種活動，有更好的聯繫和參與度。

知道學生會出現在 Kik 平台上，這使得 Kik 成為與學生聯繫非常有效的方式，在一樣使用 Sforcebot 的情況下，可以很容易地實現其他延伸應用。現在我們不是要取得商機資訊，而是要取得校園活動資訊。如下圖這個例子：



上圖中，只發送了 `Campaign` 這個關鍵字，Sforcebot 就會顯示 Salesforce 裡頭所有進行中的校園活動，除了文字訊息之外，也可以顯示圖片、開始日期與結束日期。

這樣學校就可以跟學生建立起聯繫，改善他們的參與度。