PiCamp day 4 : RESTful Web Service & JSON Tutorial

ก่อนที่จะไปรู้จักกับ JSON ขอเท้าความกันนิดนึงครับ ถ้าใครที่คุ้นเคยกับเทคโนโลยี web จะรู้จัก AJAX แน่นอน AJAX คือเทคนิคการเขียน web ด้วย Java Script เพื่อ refresh หน้า page แค่บางส่วนเพื่อที่เราจะได้ไม่ต้อง reload ข้อมูล web ทั้งหน้า เช่น เวลาเลือก drop down list ข้อมูลจังหวัด แล้วต้องการกรองข้อมูลอำเภอโดยไป load มาจาก server เฉพาะจังหวัดนั้น ถ้าเป็นการเขียนโปรแกรมแบบเก่า เราต้อง request กลับไปที่ server เพื่อ reload ข้อมูลทั้ง page แต่ถ้าเป็น AJAX ก็ไม่ต้องครับ

JSON

เมื่อก่อน เทคนิคที่ใช้ในการรับส่งกันระหว่าง browser และ server นั้นเราใช้ XML ปัญหาก็คือ XML นั้นมีการ define element ที่ยาวและมีมาตรฐานที่ซับซ้อน ทำให้ข้อมูลที่ต้องส่งมาจาก server มีข้อมูลที่ไม่จำเป็นมากมาย Douglas Crockford (http://goo.gl/5gR4W) จึงคิด format ของ JSON ขึ้นมาเพื่อให้การส่งข้อมูลเรียบง่ายขึ้น

JSON ย่อมาจาก Java Script Object Notation เป็นมาตรฐานแบบเปิดซึ่งใช้อธิบายโครงสร้างของข้อมูลแบบเดียวกับ XML แต่ลดรูปให้สั้นกว่า ลดการใช้สัญลักษณ์ที่ไม่จำเป็น

ลองดูตัวอย่างครับ อันนี้เป็นข้อมูลของลูกค้าในรูป XML

```
<person>
      <firstname>Barack</firstname>
      <lastname>Obama</lastname>
      <born>1961</born>
</person>
```

ส่วนอันนี้คือ JSON

```
{
      “firstname” : “Barack”,     // string
      “lastname” : “Obama”,
      “born” : 1961               // int
}
```

สังเกตุว่าเรียบง่ายลงมากเลย ขนาดของข้อมูลที่รับส่งกันระหว่าง client และ server ก็ลดลงด้วย

คราวนี้เรามาดูโครงสร้างของ JSON กันครับ เอกสาร JSON 1 เอกสาร (document) จะประกอบด้วย object 1 object ซึ่งจะเปิดด้วย { และปิดด้วย }

การประกาศ element หรือ attribute ของ object จะประกอบ 2 ส่วน คือ ชื่อของ element และ value ของ element คั่นด้วย : เช่น

“firstname” : “lastname”

และแต่ละ element จะคั่นด้วย , (comma)

สมมติว่าใน object person มี object address ด้วยโดย object address มี homeNo, zip และ telNo เราจะเขียนได้แบบนี้

```
{
      “firstname” : “Barack”,
      “lastname” : “Obama”,
      “born” : 1961,
      “address” : {
            “homeNo” : “123/456”,
            “zip” : 12345,
            “telNo” : 1-345-6789
      }
}
```

สังเกตุว่า address เป็น name ของ element ส่วน value จะเป็น object

ในกรณีที่ข้อมูลเป็น Array เช่น Address มีหลายที่อยู่ เราใช้ [และ] เพื่อบอกว่านี่เป็น block ของ array เช่น

```
{
      “firstname” : “Barack”,
      “lastname” : “Obama”,
      “born” : 1961,
      “addresses” : [
            {
                  “homeNo” : “123/456”,
                  “zip” : 12345,
                  “telNo” : 1-345-6789
            },
            {
                  “homeNo” : “345”,
                  “zip” : 10110,
                  “telNo” : 1-222-3333
            },
      ]
}
```

หรือในกรณีที่ข้อมูลใน JSON document เป็น array of person รูปแบบของ JSON จะเป็นแบบนี้

```
{
      “persons” : [
            {“firstname” : “Barack”, “lastname” : “Obama”},
            {“firstname” : “Steve”, “lastname” : “Jobs”}
      ]
}
```

สังเกตุว่าโครงสร้างของ JSON จะคล้ายๆ object และ class ของภาษา Java หรือ C# ถ้าใครอยากรู้ว่าจาก JSON document เมื่อเปลี่ยนเป็น class แล้วจะได้หน้าตาแบบไหน ลองไปดูที่ web นี้ครับ http://json2csharp.com เป็น C# ครับ ถ้าเป็น Java ก็ที่นี่ http://www.jsonschema2pojo.org

ตัวอย่าง website ที่เราสามารถทดสอบ JSON ได้คือที่นี่ครับ http://www.jsontest.com ใน web นี้จะมี URL และ API ให้เราทดลอง GET/POST ได้ เช่น ถ้าเราใช้ browser ทำการ browse ไปที่ http://ip.jsontest.com เราจะได้ IP ของเราเองกลับมาเป็นโครงสร้างข้อมูลแบบ JSON ครับ

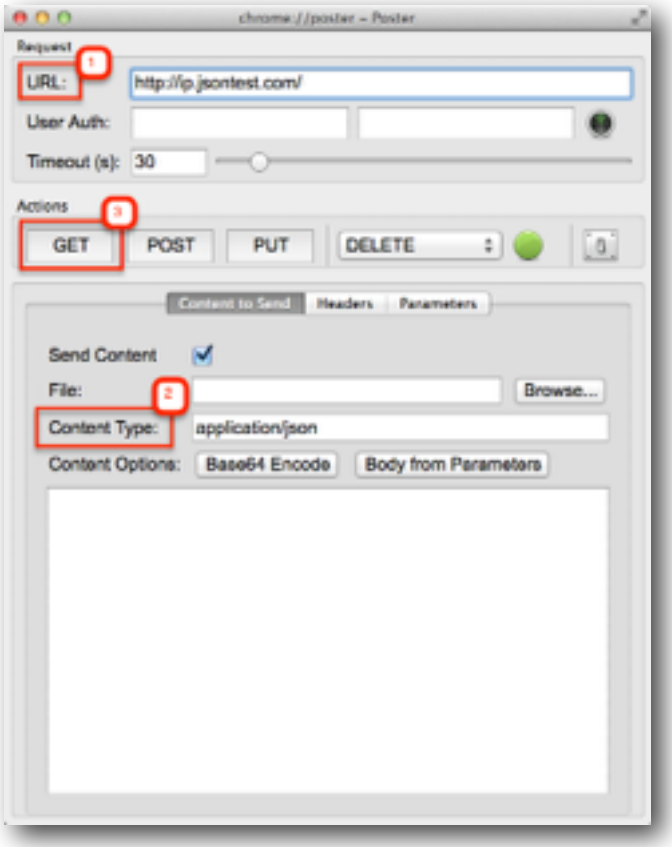
RESTful Web Service

RESTful มาจาก Representational State Transfer (หรือ REST) กำหนดโดย WC3 เพื่อเป็นทางเลือกในการ implement solution แบบ web service ด้วยเทคโนโลยีของ web ที่มีอยู่แล้ว เน้นความเรียบง่ายมากกว่า XML Web Service (รายละเอียดเพิ่มเติม ดูได้ที่นี่ http://en.wikipedia.org/wiki/RESTful)

ผมขอข้ามมาลองเล่นก่อนครับ แล้วค่อยกลับไปอ่านทฤษฎีกัน ใน tutorial นี้ผมใช้ Poster ของ Firefox ครับ ส่วน XHR Poster ของ Chrome ก็คล้ายๆ กัน

Poster และ XHR Poster เป็นเครื่องมือที่เอาไว้ GET / POST ข้อมูลไปยัง web server เพื่อทดสอบ web service หรือ RESTful โดยที่เราไม่จำเป็นต้องเขียนโปรแกรม โดย Poster นั้นจะจำลองการทำงานของ web browser ซึ่งเราจะต้องกำหนดข้อมูลใน HTTP header และ body แล้ว request ไปที่ server จากนั้นเมื่อ server response กลับมาเราก็สามารถดู header กับ body ใน HTTP Response ได้

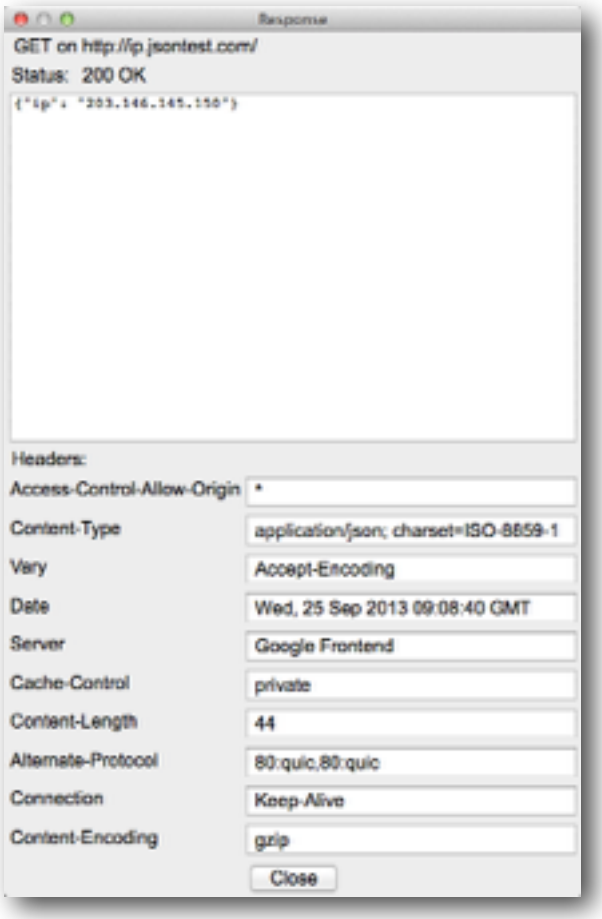
เมื่อเราเปิด Poster ขึ้นมา เราจะได้หน้าตาโปรแกรมเป็นแบบนี้



ผมจะใช้ Poster Request ข้อมูล JSON ไปยัง web jsontest.com โดยใช้ Poster ครับ โดยที่

1. ใส่ URL “http://ip.jsontest.com” ในช่อง URL
2. เปลี่ยน content type เป็น application/json
3. click ปุ่ม GET

เราจะได้ result หน้าตาแบบนี้ครับ



สิ่งที่ได้มาคือ HTTP Response โดยที่ HTTP body นั้นจะเป็น JSON object

ในกรณีที่เป็นการ POST ข้อมูลไปยัง server ก็ไม่ต่างกันครับ (POST คืออะไร ให้อ่านในภาคทฤษฎีข้างล่างครับ) แต่เราต้องใส่ JSON เป็น content ในช่อง body ด้านล่างด้วย แล้ว click ที่ปุ่ม POST แทน (ลอง POST ดูครับว่าได้อะไร)

ถ้าที่ server มีการกำหนด authentication แบบ Basic Authentication (คือจะมี popup ใน browser ให้ใส่ user name / password) เราก็กรอกในช่อง User Auth : ไว้ครับ

หลักการของ RESTful

ก่อนอื่นเรามาทำความเข้าใจ XML Web Services ก่อน หลักการของ WS ก็ง่ายๆ คือ client ทำการ request ไปที่ URL ของ server โดยส่ง XML ไปด้วยชุดหนึ่งเพื่อบอกว่าจะ request อะไร เมื่อ server ได้รับ message ก็จะส่ง content ที่ client ต้องการกลับมา ข้อมูลที่รับส่งกันจะเป็น XML ที่จะต้องมี element ถูกต้องตามมาตรฐาน web service เรียกว่า WSDL (Web Service Description Language) ซึ่งทำความเข้าใจยาก ซับซ้อน กว่าที่จะได้ข้อมูลมาก็ต้องผ่านหลายขั้นตอน

การออกแบบ Web Services จะมองว่า server ที่ให้บริการมี service อะไรบ้าง เช่น getWeather() หรือ updateWeather(Bangkok) เป็นต้น เมื่อเขียนโปรแกรม คนที่ออกแบบก็จะแปลง method ไปเป็น service ตรงๆ ซึ่งการทำแบบนี้ทำให้เราต้อง map ระหว่าง web technology และ object-oriented ทำให้การรับส่งข้อมูลระหว่าง client / server มีความซับซ้อน

แต่หลักการของ RESTful นั้นจะมองว่า server มี resource อะไรให้ มากกว่าที่จะมองว่ามี service อะไรให้ใช้ โดยใช้ URL และ HTTP Method เป็นตัวกำหนดว่า client ต้องการอะไร (ในที่นี่เราไม่ต้องสนใจว่า server ถูก implement ยังไง เพราะอยู่นอก scope ของ tutorial ฉบับนี้) เช่น ถ้าเราต้องการข้อมูล customer เราก็กำหนด URL ตรงๆ ได้เลย และ result ที่ได้ก็จะได้มาเป็นข้อมูลที่เราต้องการเลย (อาจจะเป็น XML หรือ JSON ก็ได้ ขึ้นอยู่กับการ implement ที่ server) เช่น

```
http://www.mySampleRest/persons/
```

เราจะได้ข้อมูลแบบนี้ (ย้ำว่าไม่ต้องสนใจว่าที่ server implement ยังไง อันนี้เป็นตัวอย่างว่าเราเรียกแบบนี้ เราได้ result แบบนี้)

```
{
	“persons” : [
		{“firstname” : “Barack”, “lastname” : “Obama”},
		{“firstname” : “Steve”, “lastname” : “Jobs”}
	]
}
```

แต่ถ้าเราต้องการข้อมูลแบบเฉพาะเจาะจง เช่น ต้องการข้อมูลของ person id = 1 เราก็เรียกแบบนี้

```
http://www.mySampleRest/persons/1
```

ข้อมูลที่ได้กลับมาก็จะเป็นแบบนี้

```
{
	“firstname” : “Barack”,
	“lastname” : “Obama”
}
```

พูดง่ายๆ ว่า การออกแบบ RESTful นั้นจะมอง pattern ของ URL เป็น resource แทน service ถ้าย่อยโครงสร้างของ URL จากตัวอย่าง จะมองได้แบบนี้ครับ

```
http://www.mySampleRest/persons/1
        ^               ^       ^
|  root resource | resources | resource
```

กำหนด action ด้วย HTTP Method

ใน web service แบบเดิมมันก็มี Get หรือ Update ที่ถูก implement แยกไปตามแต่ละ web method แต่ใน RESTful นั้นจะใช้ HTTP Method เพื่อบอก server ว่าเราจะทำอะไรกับ resource เหล่านั้น

HTTP Method นั้นมีอยู่แล้วในมาตรฐานของ web เช่น เวลาที่เราใช้ browser เปิด web google.com browser จะทำการ request ไปที่ server ของ google ของ method GET โดย default แต่ในมาตรฐานของ HTTP นั้นจะมีมากกว่านั้น แต่ที่นิยมใช้กันหลักๆ ก็คือ GET, POST, PUT, และ DELETE ส่วนใน web ทั่วไปก็จะใช้แค่ GET และ POST เราค่อยๆ ดูรายละเอียดกันไปทีละเปลาะครับ

ก่อนอื่น เรามาทำความเข้าใจการทำงานของ HTTP กันก่อน เอาง่ายๆ เปิด web google.com

กรณี GET

1. เมื่อเราพิมพ์URL ใน browser แล้วกด enter โปรแกรม browser จะทำการสร้าง HTTP Request ขึ้นมา ซึ่งใน HTTP Request นั้นจะประกอบไปด้วย 2 ส่วน คือ Header และ Body โดยใน HTTP Header นั้นจะมีข้อมูลต่างๆ เช่น URL ของ Server ที่เราต้องการ, ประเภทของ content, มาตรฐาน HTTP, อื่นๆ และ HTTP Method ซึ่งโดย default จะเป็น “GET” ส่วน HTTP Body จะไม่มีอะไร เพราะเมื่อระบุเป็น GET ที่ server จะไม่สนใจ body เลย
2. เมื่อ server ได้รับ HTTP Request มาจาก client ก็จะทำการแกะเอา header ออกมาดูว่า client จะเอาอะไร ในกรณีนี้ browser request มาเป็น GET เพราะฉะนั้นก็ขึ้นอยู่กับว่า server จะส่งอะไรกลับมา โดยปกติก็จะส่ง html document กลับมา แต่การส่งกลับมานั้น server จะส่งมาเป็น HTTP Message เหมือนกันแต่จะเรียกว่า HTTP Response ซึ่งใน Response จะมี HTTP Header เหมือนกัน แต่ content ของ html จะอยู่ใน HTTP body
3. เมื่อ browser ได้รับ HTTP Response กลับมาจาก server ก็จะทำการแกะ header ออกมาเพื่อ verify และเอา HTTP body ที่ได้กลับมา มาแสดงผลใน browser ก็ถือว่าจบกระบวนการ

ขั้นตอนทั้ง 3 ข้อนี้เป็นการ request / response แบบ GET กรณีที่เปลี่ยนจาก web ปกติไปเป็น RESTful นั้น การ request ไม่ต่างอะไรจาก web ปกติเลย คือระบุ URL, กำหนด HTTP Method เป็น GET แล้ว request ไปที่ server ที่ให้บริการ เราก็จะได้ข้อมูล JSON มาใช้ต่อได้เลย

จุดเดียวที่อาจจะต่างกันก็คือ ในกรณีที่ server ใช้ URL เดียวกัน แต่แยกแยะว่าข้อมูลที่ส่งกลับมาให้ client นั้นจะเป็น html หรือ JSON โดยการกำหนดจาก client ก็คือการระบุ content type ว่าเป็น `text/html` หรือ `application/json` ด้วยหลักการแบบนี้ เราสามารถออกแบบ RESTful ได้โดยไม่ต้องแยก URL เลย

กรณี POST

ตัวอย่างเช่น เราต้องการ search google ด้วย keyword และ google กำหนดว่าเราต้องใช้ POST method ในการเขียน web นั้นเราสามารถเขียน action ของปุ่มได้ว่าถ้า click ที่ปุ่ม “Search” ให้เอาข้อมูลใน text box ส่งไปที่ server ด้วย scenario จะเป็นแบบนี้ครับ

1. จากหน้า search ของ Google เราพิมพ์ keyword ลงไปในช่อง search แล้ว click ปุ่ม search ซึ่งกำหนดไว้แล้วว่าให้ browser request ไปที่ server แต่เป็น POST Method
2. browser จะทำการสร้าง HTTP Header เหมือนเดิม แต่คราวนี้จะกำหนด HTTP Method เป็น POST แทน

3. ในส่วนของ HTTP Body นั้น browser จะทำการสร้างโครงสร้างข้อมูลที่ server เข้าใจ จากนั้นก็จะ pack ทั้ง header และ body เป็น HTTP Request แล้วส่งไปที่ server
4. เมื่อ server ได้รับข้อมูลมา server ก็จะรู้ว่า คราวนี้มาเป็น POST จาก HTTP Method แสดงว่าใน HTTP Body จะต้องมีข้อมูลอะไรบางอย่างส่งมาด้วย ก็จะทำการอ่านข้อมูลใน body แล้ว validate ว่าข้อมูลถูกต้องไหม และนำไปประมวลผล (ในกรณีของเรา คือเอา keyword ไป search)
5. เมื่อได้ result แล้วก็เหมือนเดิม คือทำการสร้าง HTTP Response เพื่อส่งกลับไปที่ client โดยสร้าง header ขึ้นมาและ stamp status ลงไปว่า OK (result = 200) แล้วเอาผลการ search ใส่ลงไปใน HTTP body ในรูปของ html แล้วส่ง HTTP Response กลับมา

กลับมาที่ RESTful กัน หลักการก็เหมือนกับตัวอย่าง GET/POST ที่ว่ามา แต่จะแตกต่างกันที่ content ที่ใช้รับส่งกันไม่ใช่ HTML แต่เป็น JSON (อาจจะเป็น text หรือ XML ก็ได้) และแยก action ว่าเราอยากทำอะไรกับ resource ที่ระบุใน URL เช่น

URL : http://www.mySampleRest.com/persons

GET : หมายถึง get เอา person ทั้งหมด
POST : หมายถึง การ Insert เอา person คนใหม่เข้าไปในระบบ (body ก็เป็น JSON document)
PUT : หมายถึง การ update ข้อมูลของ person ตาม JSON ที่เราระบุใน HTTP body
DELETE : หมายถึง delete ข้อมูล person ตาม JSON ที่เราระบุใน HTTP body

ด้วยกลไกของ HTTP แบบนี้ เราก็สามารถนำมาใช้กับ RESTful Web Service ได้แล้วโดยไม่ต้องกำหนดมาตรฐานของ content ที่รับส่งกันมากมาย ข้อมูลที่รับ/ส่งไปมาในรูป JSON ก็เป็น text ธรรมดา เราสามารถสร้าง text ในรูป JSON เพื่อ request ไปที่ server ได้เลย หรือในภาษาเขียนโปรแกรมสมัยใหม่ก็มี library ที่รองรับการแปลงข้อมูลไปมาระหว่าง object และ JSON อยู่แล้ว การเขียนโปรแกรมเพื่อเรียก web services ก็ทำได้ง่ายขึ้นเมื่อเทียบกับ XML Web Service ครับ