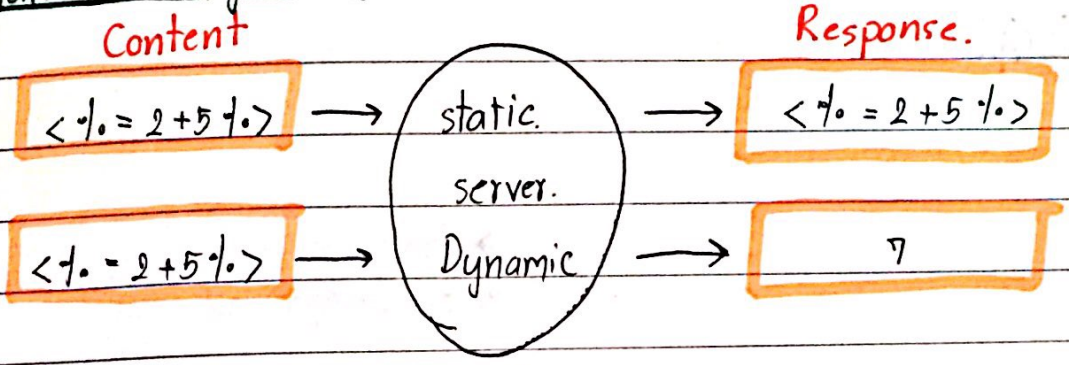
Chapter 5.
Web Server.

Web Server ยอดนิยม.
- Apache
- IIS
- nginx

Web Server

- ใช้ในการประมวลผล แล้วตอบกลับไปยังผู้ใช้
- Support server-side scripting คือ จะมีการประมวลผลฝั่ง server ที่ละบรรทัด (scripting)
- ใช้ Protocol (Http) เป็นขั้นในการติดต่อสื่อสาร

Static vs. Dynamic



- Static : Content ยังไง ก็ Response อย่างนั้น.
  - ↳ ไม่สามารถมีตัวแปรได้ / คำนวณไม่ได้
- Dynamic : ใช้ในการคำนวณฝั่ง server.

Dynamic

ใช้ในการคำนวณฝั่ง server.

- CGI เป็นชุดที่ช่วยประสานงานระหว่าง web server กับ program ผลลัพธ์ประมวลเป็นแบบ Dynamic Content แต่ทำงานได้ไม่เต็มประสิทธิภาพ. ทำงานได้ 1 request ต่อ 1 process ซึ่งจะทำงานช้า ทรัพยากรมาก.
- FastCGI ทำงานได้ดีมากกว่า 1 request ต่อ 1 process
- SCGI.
- Platform
  - Microsoft IIS : ISAPI (Internet Server API).
  - Java : Servlet Container.
  - Ruby : Rack
  - Perl : WSGI (Web Server Gateway Interface).

Date : / /

Note :

CGI

- เป็นชุดเชื่อมต่อระหว่าง Web server กับ program ในการสร้าง Web content
- server จะ run script แต่ละอันโดยแยก process กันไป แล้วส่งผลลัพธ์ให้ Client
- สิ่งที่ใช้ในการส่งผลลัพธ์ให้ Client จะเป็น environment variable
  ex. Query_string.
- CGI scripts สามารถเขียนโปรแกรมด้วยภาษาใดก็ได้ แต่ต้องบอก web server ก่อน.
  ex. Perl, Python

Node as a Script

<< Apache2 configuration file >>

```
<Directory /var/www/html/cgi>
    Options      +ExecCGI +SymLinksIfOwnerMatch
    Action       node-script  /cgi-bin/node-cgi
    AddHandler   node-script  .nd
</Directory>
```

/cgi-bin/node-cgi → เป็นตัว run.

.nd → file ที่อยู่ใน folder ของมาลงท้าย .nd.

ex. ชื่อ file.

<< CGI Script (test.nd) in JavaScript >>

```
for(k in env){
   writeLine(k + "=" + env[k] + "<br/>");
}
```

- Create a Web Server.

```
var http = require('http');
var server = http.createServer(function(req, res){
   res.writeHead(200, {'Content-type': 'text/plain'});
   res.end('Hello world\n');
});

server.listen(8000);
console.log('Server is ready!');
```

เปิด port 8000

Date: / /

Note:

Express Routing

```
var express = require('express');
var app = express();

app.get('/', function(req, res){
  res.send('Hello world')
});

app.listen(8000);
```

- Route เป็นการจัดกลุ่ม.
  - URI
  - Http request method (Get, post)
  - handler.
- structure.

app.Method(path, [callback...], callback)

(Method → ชื่อ Method)

- app เป็น object ของ express
- สามารถเรียกใช้ Method ได้หลากหลาย.
  - Get ข้อมูลที่จะฝากอยู่ใน Query string ส่งข้อมูลได้ไม่เกิน 255 bytes ไม่ปลอดภัย เช่นการบันทึกรหัสผ่าน สะดวก และใช้งานง่าย.
  - Post จะเป็น data ของ http header ขนาดได้เยอะกว่า get และปลอดภัยกว่า.
- path คือ url หรือ Callback function.

Express Middleware

- Express เป็น Middleware ตัวหนึ่ง.
- Middleware: เข้าถึง request object ได้ ช่วยตอบกลับผ่าน response object และ response object สามารถแก้ไขได้ผ่าน middle ware
- Middleware สามารถ...
  - สามารถ Execute any code ได้
  - เปลี่ยนแปลง request และ response object ได้.
  - สั่งจบ request-response cycle ได้
  - สั่งให้โยง / เชื่อมต่อไปยัง middleware ตัวอื่น. → middleware.

Middleware จะใช้ "app.use"

```
// a middleware with no mount path; gets executed for every
request to the app
app.use(function (req, res, next) {
  console.log('Time:', Date.now());
  next();
});

// a middleware mounted on /user/:id; will be executed for any
type of HTTP request to /user/:id
app.use('/user/:id', function (req, res, next) {
  console.log('Request Type:', req.method);
  next();
});
```

- next → ส่งต่อไปยัง middleware ตัวถัดไป
- Date.now() → จะตอบกลับอะไรบ้าง.
- 'Time:' → server ส่งมา
- :id → เป็นตัวแปร ใน url ไม่ต้องใส่ :

Date : / /

Note :

Mon Tue Wed Thu Fri Sat

Built-in / 3rd party middleware

▶ Only 1 built middleware — อ้างอิงได้ folder เดียว.

- express.static — เป็นการบอกว่า folder ไหนที่จะให้เข้าถึงได้

```
app.use(express.static(`public`));
```

อ้างถึง resource (ไฟล์ที่ใช้กับ-
-การสั่งการ).

▶ Useful 3rd party middleware

การเก็บข้อมูลที่สามารถเรียกใช้ได้จากทุกๆ page ทุกๆหน้า

- **Cookie-parser**

  เก็บที่ Client เป็นการยืนยันตัวตนของ Client ตัวนั้น เช่น สมัครสมาชิกต่างๆ ที่ Client จะรู้ว่า Client คนนี้ชื่ออะไร.

- **Express-session**

  จะเก็บที่ server เช่น เก็บชื่อไอที่ Server โดยไม่รู้ว่าชื่อนั้นเสื่อ Client คนไหน.

- **Body-parser**

  การส่งค่าจาก Client ไป Server เช่น การรับค่าเพื่อนำไปบวกเลข.

Full example : Adding

server.js

```
var express = require('express'),
    app = express(),
    bodyParser = require('body-parser');

var urlencodedParser = bodyParser.urlencoded({ extended: false });
app.use(express.static(__dirname + '/public'));

app.post('/add', urlencodedParser, function(req, res){
   var result = parseInt(req.body.a) + parseInt(req.body.b);
   res.send('Result = ' + result);
});

app.listen(8000);
```

คู่ที่รับมาเป็น string ต้องแปลงเป็น integer.

```
>> npm install express body-parser
```

public/form.html

```
<html>
<head>
<title>Adding Form</title>
</head>
<body>
   <form action="/add" method="post">
      A: <input type="number" name="a"/><br/>
      B: <input type="number" name="b"/><br/>
      <button type="submit">Add</button>
   </form>
</body>
</html>
```

Date: / /

Note:

Mon Tue Wed Thu Fri Sat Sun

Web Session Tracking

- HTTP is a "stateless" protocol → ไม่มีสถานะ
  ทุกครั้งข้อมูลไปหน้าอื่นๆ จะไม่มีข้อมูล / สถานะ ดังนั้นจะทำได้ 4 อย่าง.
- Session Tracking.
  1. URL Rewriting.
  2. Hidden From Fields ต้องใส่ hidden ในทุกๆหน้า
  3. Cookies. เก็บข้อมูลที่ Client
  4. Sessions. เซ็ตครั้งเดียวแล้วใช้ที่ไหนก็ได้ เก็บข้อมูลที่ server.

Cookies (on Client)

เก็บที่ Client

ช่วยในการเข้ารหัส.

```
app.use(cookieParser('keyboard cat'))

app.get('/ck_get', function(req, res) {
   res.send(req.cookies) → cookies server จะเป็นคนสร้าง แต่เมื่อสร้างแล้วจะนำไปเก็บที่ Client
})

app.get('/ck_set', function(req, res) {
   res.cookie('a', 10) ↘ สร้าง Cookie ชื่อ a มีค่า 10
   res.send('ok')
})
```

Cookies สร้างได้ทีละตัว.

Sessions (on Server). เก็บที่ Server.

```
app.use(session({ secret: 'keyboard cat', cookie: { maxAge: 60000 }}))
app.use(function(req, res, next) {
   var sess = req.session
   if (sess.views) {
      sess.views++
   } else {
      sess.views = 1
   }
})
```

Date: / /

Note:

Express Template Engine

เอาข้อมูลที่ใน template file มาแสดงผล.

Server → เก็บ database

server.js

```
app.set('views', './views')
app.set('view engine', 'ejs')

app.get('/fruit', function(req, res){
  res.render('fruit', {fruits: ['banana', 'apple']})
})
```

ชื่อfolder.

กำหนดรูปแบบในการแสดง template.

views/fruit.ejs

```
<ul>
<% fruits.forEach(function(fruit){ %>
  <li><%= fruit %></li>
<% }); %>
</ul>
```

ส่งข้อมูลไปใช้ใน file fruit.ejs

ตัวแปรที่จะนำไปใช้ใน file fruite.ejs

Scanned by CamScanner

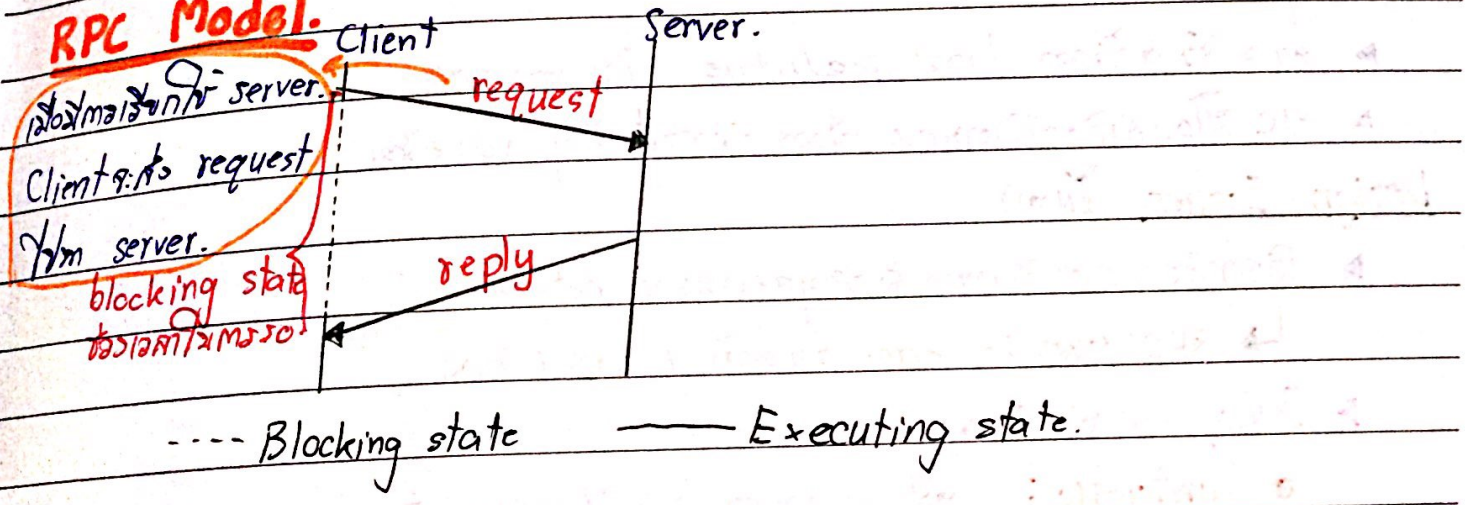
Note:

Chapter 6.
RPC & REST

Remote Procedure Call (RPC)

- สามารถเรียกใช้ function ตามระยะไกล เป็น level ระดับสูง ทำหน้าที่ติดต่อสื่อสารระหว่าง client - server.
- ช่วยให้นักพัฒนาโปรแกรมสร้างระบบขนาดใหญ่
- Example: File service, Authentication service.

RPC Model.



Characteristics

- function ที่ถูกเรียก จะ run อยู่ในเครื่องอื่น.
- function ที่ถูกเรียก จะไม่แชร์ address space.
  - passed by values : pass ได้แบบเดียว คือ passed by values เท่านั้น copy ค่า แล้วส่งไป
- จะ run อยู่บน server (environment ของ server) ซึ่งจะกิน Mem ของ server
  - server เข้าถึง client ไม่ได้
- โปรแกรมเมอร์จะไม่เห็นข้อมูลที่ส่งผ่าน network

Date: / /

Note:

Mon Tue Wed Thu Fri Sa

Features (คุณสมบัติของ RPC).

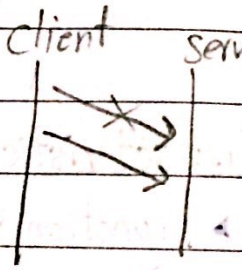
- Simple call Syntax.
- มีแต่ prototype ก็เรียกใช้ได้.
- RPC มีการกำหนด interface ได้
- ใช้งานง่าย.
- มีประสิทธิภาพ.
- สามารถติดต่อกับเครื่องอื่นได้

Limitations (ข้อจำกัดของ RPC)

- ส่ง address ไม่ได้ ทำได้แค่ copy ค่าแล้วส่งไป เพื่องากเข้าถึง environment ของผู้เรียกไม่ได้
- ความเร็ว จะช้ากว่า local procedure กินเวลามากกว่า
- RPC มีโอกาสเกิดข้อผิดพลาด เพื่องากมีการส่งข้อมูลไปยังเครื่องอื่น

Design Issues (ปัญหา).

- เกิดข้อผิดพลาดเพื่องากส่งข้อมูลผ่านทาง network.
  - RPC ตรวจสอบ error จากค่าที่ return กลับมา.
- Transparence (การออกแบบ).
  - Syntactic : เหมือนอยู่แบบการ call มี argument
  - Semantic : เหมือนกันทุกอย่าง ซึ่ง RPC จะทำไม่ได้

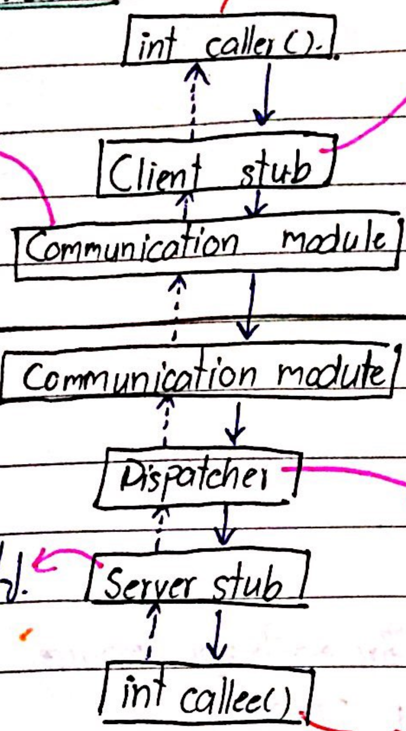


- Guarantees (การันตี).
  - Retry request : client ส่ง request ไป แต่ไม่ได้รับ message ตอบกลับ Client ก็ต้องส่งใหม่ เผื่อ: อาจสูญหายจากสองทาง
  - Duplicate filtering: server ทำ หาก server ได้รับแล้ว แต่ Client มีการส่งซ้ำ server ต้องทำ Duplicate filtering เพื่อกันไปอันนึง
- Retransmission of replies: กรณีที่ server ตอบกลับโดยส่ง request แต่ client ไม่ได้รับ. clientจึงส่ง retry request อีกรอบ ซึ่งจะเก็บ history ของการ reply ไว้ได้

Date: / /

Note:

RPC Components.



ผู้เรียกใช้

ส่งให้ server ผ่าน client stup
ซึ่ง Client stub ทำหน้าที่...
- ▶ ตรวจสอบ syntax ของการเรียกใช้
- ▶ แปลงข้อมูลให้อยู่ในรูปแบบ RPC เพื่อให้ protocal เข้าใจกัน.

ส่ง request ไปยัง communication ของ server.

เรียกใช้ server stube บอกว่า methode ที่เรียกใช้เป็น method อะไร.
แปลงข้อมูลจากรูปแบบมาตรฐานกลับไปเป็นรูปแบบของ server และส่งให้ run method นั้นในฝั่ง server.

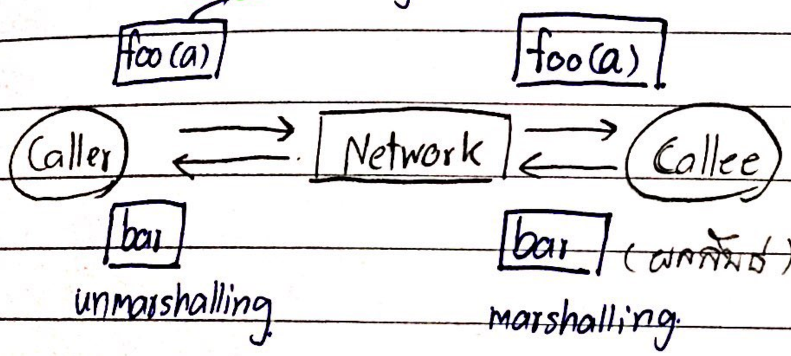
ส่ง value กลับไป.

ผู้ถูกเรียกใช้

Marshalling

marshalling แปลงให้อยู่ใน CDR format แล้วส่งไปที่ network.

CDR = data respresentation.



- ▶ Marshalling. : กระบวนการที่ใช้ในการแปลงข้อมูลรูปแบบหนึ่งเป็น CDR
- ▶ Unmarshalling : แปลงจาก CDR เป็นรูปแบบ form ของตัวเอง.

JSON-RPC

- ▶ เป็น RPC ที่เข้ารหัสใน JSON.
- ▶ เป็น protocal ที่ง่าย.
- ▶ ช่วยในการแจ้งเตือน (ข้อมูลที่ส่งไปยัง server ที่ไม่มีการตอบกลับ).

Date : / /

Note :

Mon Tue Wed Thu Fri

Adding

server.js

```
var rpc = require('json-rpc2');
var server = rpc.Server.$create();

function add(args, opt, callback) {
    callback(null, args[0] + args[1]);
}

server.expose('add', add);
server.listen(8000, 'localhost');
```

```
>> npm install json-rpc2
```

เรียกใช้ method add.

client.js

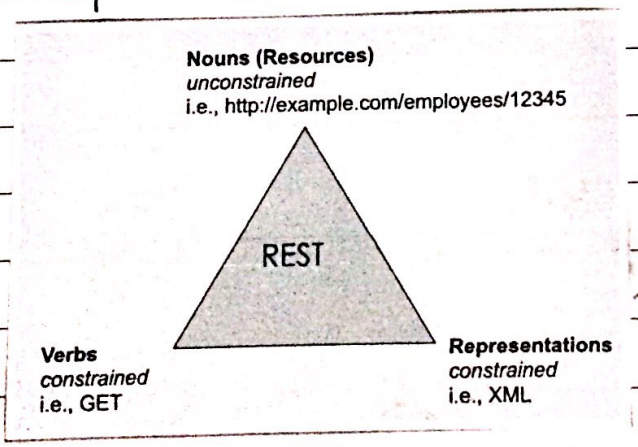
```
var rpc = require('json-rpc2');
var client = rpc.Client.$create(8000,
  'localhost');

// call the function on the server
client.call('add', [1, 2],
  function(err, result) {
    console.log('1 + 2 = ' + result);
  }
);
```

REST and HTTP

- เป็นการ call procedure ที่เครื่องหนึ่งอยู่ในรูป resource.
- เป็น Remote Procedure Call (RPC).
  - URI addressable ใช้อ้างอิง
- Web server สามารถใช้ได้

Main Concepts.



Resources

- เป็น key abstraction ของข้อมูลใน REST
- เป็นการทำ conceptual mapping ไปยังชุดของการดำเนินการ.
- การใช้งาน resource จะใช้ผ่าน Http protocal โดยผ่าน URI (URI in HTTP protocal).
  เข้าผ่าน url ใน แต่ละ method.

Date:

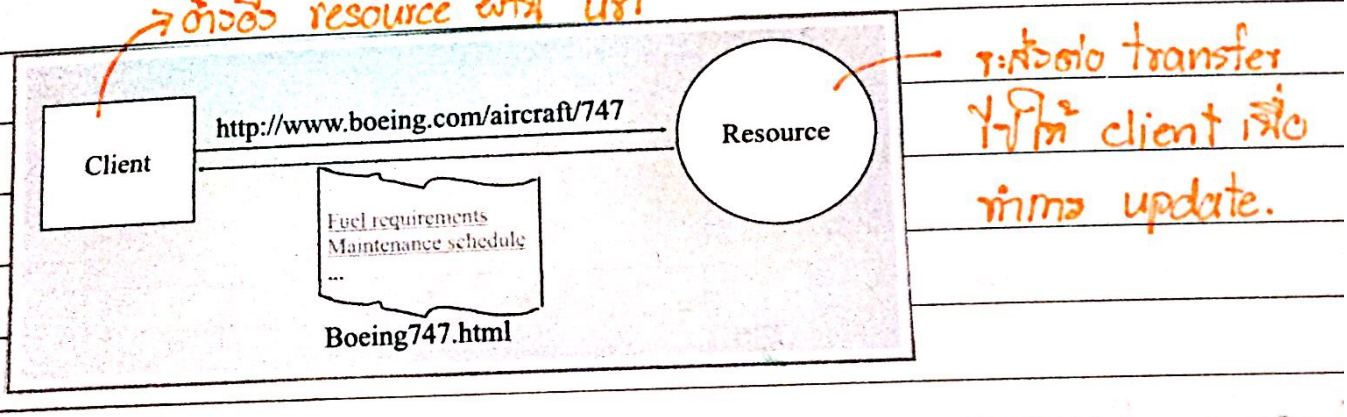
Note:

Verbs

1. HTTP GET สำหรับการดึงข้อมูล.
2. HTTP POST สำหรับการสร้างข้อมูล.
3. HTTP PUT สำหรับการแก้ไขข้อมูล.
4. HTTP DELETE สำหรับลบข้อมูล.

Representations (แสดง).

- การที่ข้อมูลจะแสดง คือ กลับไปยัง client เพื่อนำเสนอ.
- Two main formats:
  - Java Script Object Notation (JSON).
  - XML
- ตาราง: สี่ เหลี่ยม + การแสดงของข้อมูลที่ขอกัน



Example: REST for bears.

Route	HTTP Verb	Description
/api/bears	GET	Get all the bears.
/api/bears	POST	Create a bear.
/api/bears/:bear_id	GET	Get a single bear.
/api/bears/:bear_id	PUT	Update a bear with new info.
/api/bears/:bear_id	DELETE	Delete a bear.

Date : / /

Mon Tue Wed Thu Fri

Note :

Example : Create a bear.

```
var express = require('express');
var app = express();
var router = express.Router();
var bodyParser = require('body-parser');

var bears = [];

router.route('/bears')
  .post(function(req, res) {
    var bear = {};
    bear.name = req.body.name;
    bears.push(bear);
    res.json({ message: 'Bear created!' });
  });

// all of our routes will be prefixed with /api
app.use('/api', bodyParser.json(), router);
app.listen(8000);
```

Angular $http.

```
<body ng-controller="BearCtrl">
  <input type="text" ng-model="bear.name">
  <input type="number" ng-model="bear.weight">
  <button ng-click="add()">Add</button>
</body>

<script type="text/javascript">
  angular.module("bearApp", [])
  .controller("BearCtrl", function($scope, $http){
    $scope.add = function(){
      $http.post("/api/bears", $scope.bear).
        success(function(data){
          console.log(data.message);
          $scope.bear = {};
        });
    }
  })
</script>
```

Angular Resource

```
<script type="text/javascript">
  angular.module("bearApp", ["ngResource"])
  .controller("BearCtrl", function($scope, $resource){
    var Bear = $resource('/api/bears/:bear_id',
        {bear_id: '@id'});

    $scope.add = function(){
      var givenBear = new Bear($scope.bear);
      givenBear.$save(function(data){
        console.log(data.message);
        $scope.bear = {};
      });
    };
  })
</script>
```

Date: / /

Mon Tue Wed Thu Fri Sat

Note:

```
<body ng-controller="BearCtrl">
  <ul>
    <li ng-repeat="bear in bears">{{bear.name}}</li>
  </ul>
</body>

<script type="text/javascript">
  angular.module("bearApp", ["ngResource"])
  .controller("BearCtrl", function($scope, $resource){
     var Bear = $resource('/api/bears/:bear_id',
       {bear_id: '@id'});
     $scope.bears = Bear.query();
  })
</script>
```

Date : / /

Mon Tue Wed Thu Fri

Note :

Chapter 7. Web Security.

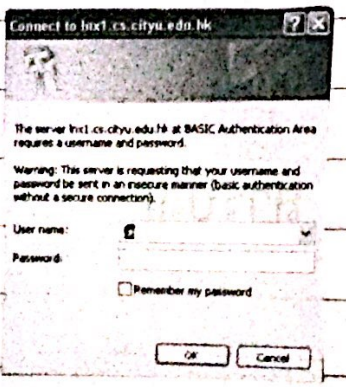
Major Security issues

- **Authentication :** การพิสูจน์ว่าคือคนๆ นั้นหรือป่าว เช่น สายนิ้วมือตรงกัน รู้รหัสผ่าน (การพิสูจน์ตัวตน)
- **Authorization :** อนุญาตให้คน ~~xxx~~ คนนี้ เข้าถึง หรือใช้งานอะไรได้บ้าง (กำหนดสิทธิการเข้าถึง) ซึ่งเชื่อมกับ Access control
- **Access Control :** resource อันไหนที่ให้ user คนไหนเข้าถึงได้บ้าง เช่น staff แก้ไขข้อมูลได้บางส่วน, Admin แก้ได้ทั้งหมด.
- การส่ง data ผ่าน network ที่ปลอดภัย ต้องทำการ "Encryption" โดยการ SSL (Secure Socket Layer)

Authentication

- มีการเก็บตัว user ID จาก user และมีการ login ซึ่งต้องตรวจสอบว่า ID ที่ login ตรงกับ ID ที่ลงทะเบียนหรือไม่
- ซึ่งข้อมูลจะส่งผ่าน HTTP หรือ HTTPS ก็ได้ โดย HTTPS จะส่งผ่าน SSL
- ชื่อ user ID & password ต้อง match กัน
- **Keep track :** เมื่อ login ผ่านแล้ว จะทำการเก็บบันทึก ข้อมูล เรียกว่า "Keep track" ซึ่งอาจจะใช้ session / cookie ช่วย.

Basic Authentication



browser จะส่งขึ้นมา.

- Basic ส่งได้แค่ username, password ตามที่ browser ขึ้นมา.
- ไม่สามารถปรับเปลี่ยน GUI ได้ ใส่ได้แค่ username, password

Date: / /

Note:

Form-based Authentication

▶ อยากได้ form แบบไหน สร้าง form เองได้

Basic vs. Form-based

Basic	Form-base
▶ รับ username, password เท่านั้น. !!	▶ แก้ไขแต่งออกแบบ อยากได้อะไรก็ออกแบบได้เอง.
▶ ใช้ HTTP Authentication header ในการส่ง username, password.	▶ ส่งผ่าน data.

Basic password scheme

▶ Hash function h: string → strings.

ใช้ในการเข้ารหัส เป็นการคำนวณ.

ทำย้อนกลับเพื่อหา password ไม่ได้.

▶ เก็บ password เป็น file h (hash).

▶ ไม่เก็บ password ในเครื่อง / harddisk.

Secure Sockets Layer (SSL)

▶ https ใช้ port 443 , http ใช้ port 80.

▶ SSL จำเป็นต้องมี...

- เมื่อมีการ login เพื่อใช้ในการป้องกัน password โดนดักระหว่างการส่งใน network.
- ข้อมูลของพวกบัตรเครดิต จำเป็นต้องมี SSL.

Date: / /

Note:

Use of an SSL Certificate

- SSL ต้องใช้ Certificate.
- Certificate. เป็นการยืนยันว่า public key เป็นของคนๆ นี้จริงๆ ซึ่งจะมี Certificate Authority (CA) เป็นสื่อกลาง.
- Hybrid cryptosystem. ใช้การเข้ารหัสทั้ง public และ key เดียว. session key. คือ เริ่มต้นใช้ public & private key ก่อน สร้าง session key แล้วจากนั้นใช้การสื่อสารแบบ key เดียว.

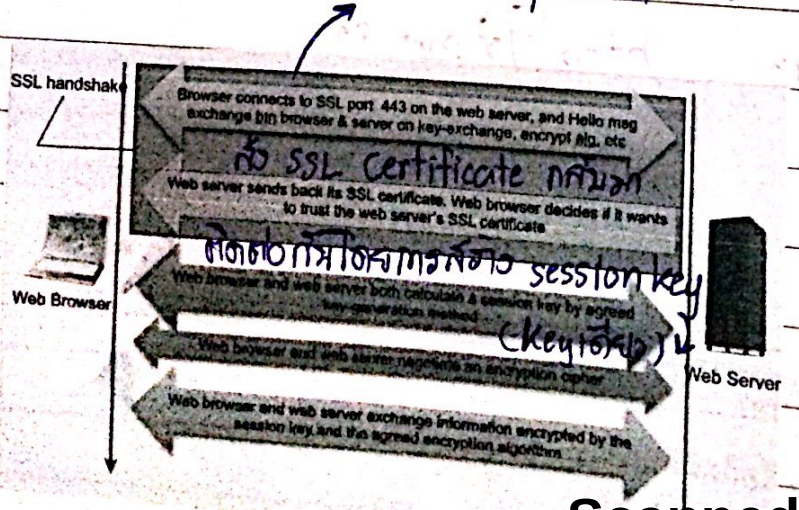
ข้อเสียของ public & private key!

- public & private key ใช้ความยาว key ที่ยาวกว่า key เดียว ที่จะต้องความปลอดภัยเท่ากัน
- กินพลังงานคอมฯ เยอะ: และใช้เวลาในการ process นานกว่า.

A Sample Certificate เก็บอะไรบ้าง...

- version.
- ผู้ออก CA
- ออกเมื่อไร.
- ออกให้ใคร.
- แนบด้วยสาธารณะ.

How SSL work? เชื่อมต่อผ่าน port 443.



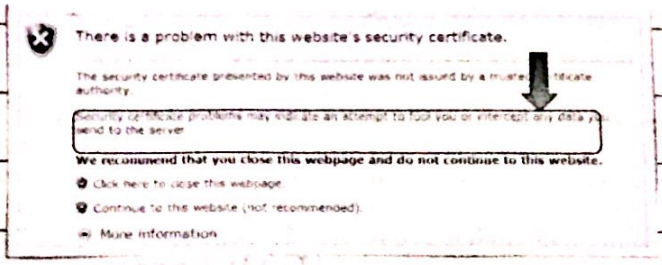
Date: / /

Note:

Mon Tue Wed Thu Fri Sat Sun

CA Root Certificate

- Certificate ที่อยู่บนสุด ที่เอาไว้ browser ต้องเชื่อถือ.



OAuth

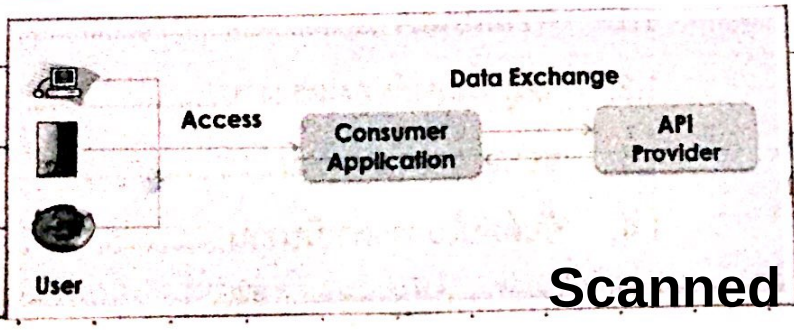
- เป็น protocal มาตรฐาน แบบหนึ่งที่ให้มี การอนุญาตในการเข้าถึงข้อมูลแบบปลอดภัย แต่ง่าย.
- ข้อมูลจะง่าย แต่ปลอดภัย
- อนุญาตให้ผู้ใช้บริการสามารถให้สิทธิเข้าถึงแก่ user.
- ใช้กับ HTTP protocal แต่ไม่ support กับ protocal อื่นๆ

Why OAuth?

- ยื่อหาปุ่ม ของปี แต่ออกแบบมาให้ใช้กับอุปกรณ์ไกลศัพท์ แต่ desktop application
- สามารถกำหนดสิทธิในการเข้าถึง resource ต่างๆได้ แต่กำหนดการร้องขอต.
  - Facebook's Graph API
  - Foursquare.
  - Github
  - Google.
  - Salesforce
  - Windows Live

} ธุรกิจที่ใช้ OAuth.

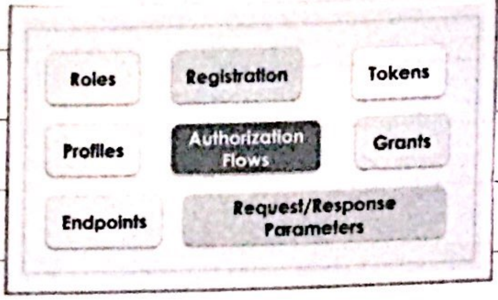
Introduction



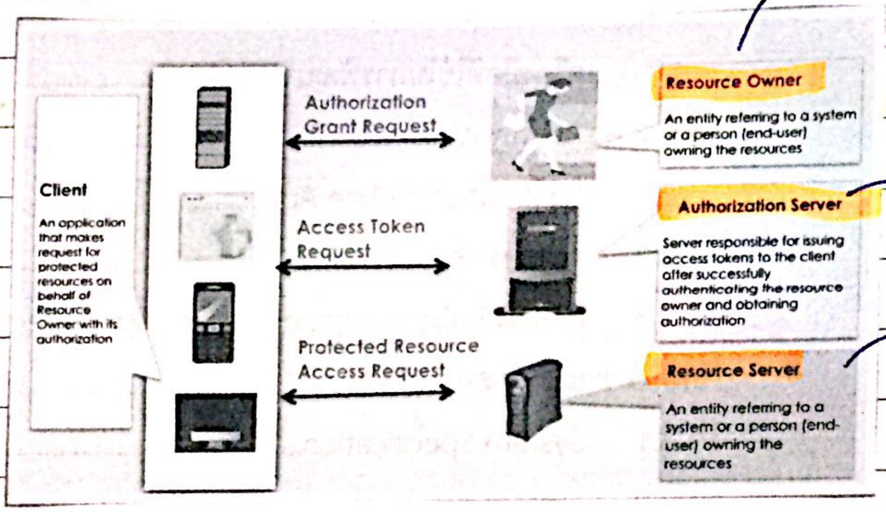
Note :

Mon Tue Wed Thu Fri

Building Blocks



Roles



เจ้าของ Resource
เช่น เจ้าของ facebook.

Server ที่ทำการพิสูจน์ตัวตน ยอมให้เข้าถึง resource

เป็นคนเก็บ resource
เช่น facebook.

Client Registration

- ~~OAuth~~ ต้องลงทะเบียนกับ server ก่อนที่จะทำการร้องขอ API ได้

Client Profiles

- การเข้าถึง จะต้องเรียก API จาก resource server.
- จะไม่เก็บ password ของ user

Tokens

- เป็นรหัสผ่าน เป็นข้อมูลตัวเลข / ข้อความ. เปรียบเสมือนชิป เช่น ชิปในบัตรขึ้นรถไฟฟ้า
- token แบ่งเป็น 2 แบบ.
  - Access Token: เป็น token ในการเข้า resource นั้นๆ
  - Refresh Token: ใช้ในการขอ Access Token.

Date: / /

Note:

Mon Tue Wed Thu Fri Sat Sun

Grant Types

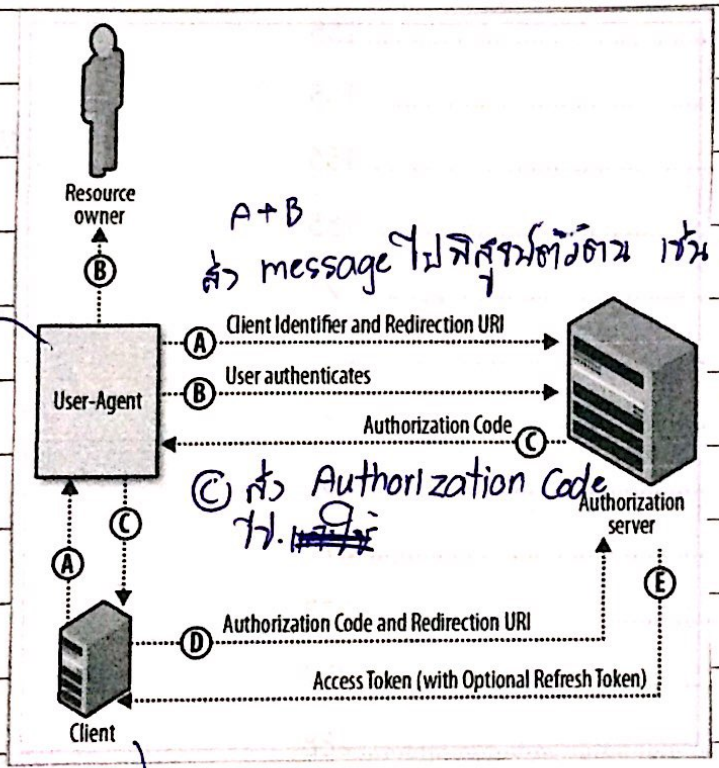
▶ เป็นชนิดที่ใช้กำหนดการเข้าถึง.

Authorization Flows.

1. Server Side Web Application Flow.
2. User Agent (Browser) Application Flow.
3. Resource Owner Password Flow.
4. Client Credentials Flow.

Server-side Web Application Flow

การขอเข้าถึง จะต้องมีการพิสูจน์ตัวตน กับ Authorization server.



เช่น browser

A+B ส่ง message ไปพิสูจน์ตัวตน เช่น เลสามี pop up ขอเข้าถึงสูปภาพ. และต้องใส่ username, password. ส่งไปให้ Authorization server

© ส่ง Authorization Code ไป. ~~ที่ใช้~~

(D) ใช้ Authorization code และ Redirection URI ส่งไปให้ Authorization server เพื่อเข้าไปขอสิทธิ์

ออกเข้าถึง resource owner

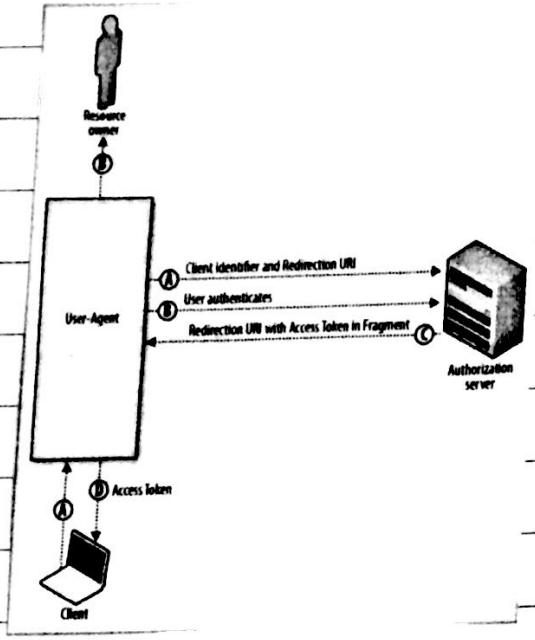
Application Web server.

(E) และจะมี Access Token กลับมา ซึ่ง Access Token จะมีช่วงเลสามีนการเข้าถึง resoure หมดอายุ Token จะหมดอายุ

Date : / /

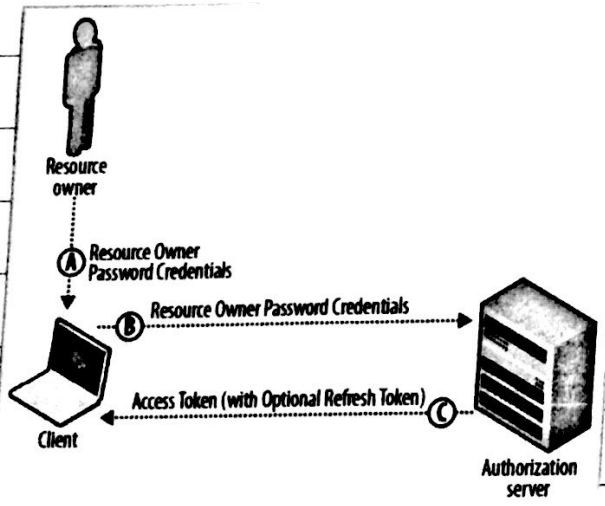
Note :

Client-Side Web Application Flow



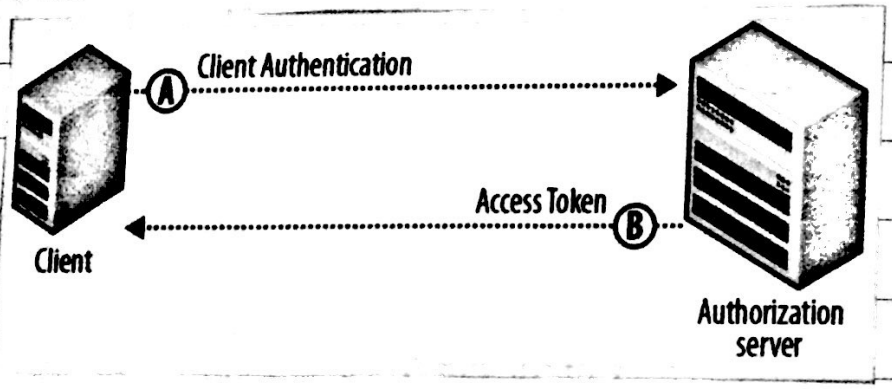
จะได้ Token เลย
ไม่ต้องส่งขอ Token

Resource Owner Password Flow



Client จะได้ password ไปใช้เลย.

Client Credentials Flow



Date: / /

Note:

Mon Tue Wed Thu Fri Sat Su

Passport

- เป็น middleware ตัวหนึ่งใช้ทำ authentication ของ node.
- login ครั้งเดียวใช้งานได้ตลอด.

Facebook as a Passport Provider.

- ใช้ passport ด้วย.
- ในการสร้าง app ต้องมี App ID และ App Secret.