Chương 5

Web API

CT275 – CÔNG NGHỆ WEB

Mục tiêu

Giới thiệu Web API và kiến trúc REST API

Nội dung

- Giới thiệu Web API
- Cơ bản về REST
- Chứng thực người dùng REST API
- Cài đặt và tiêu thụ REST API

Giới thiệu Web API

MVC Architecture Introduction

API (Application Programming Interface)

- "là một tập các hàm và thủ tục dùng để tạo ra các ứng dụng, cho phép các ứng dụng truy xuất đến các chức năng, dữ liệu của hệ thống, của các ứng dụng hay các dịch vụ khác"

(Oxford English Dictionary)

- "một giao diện hay thành phần trung gian cho phép một ứng dụng giao tiếp với các ứng dụng khác"

(Investopedia)

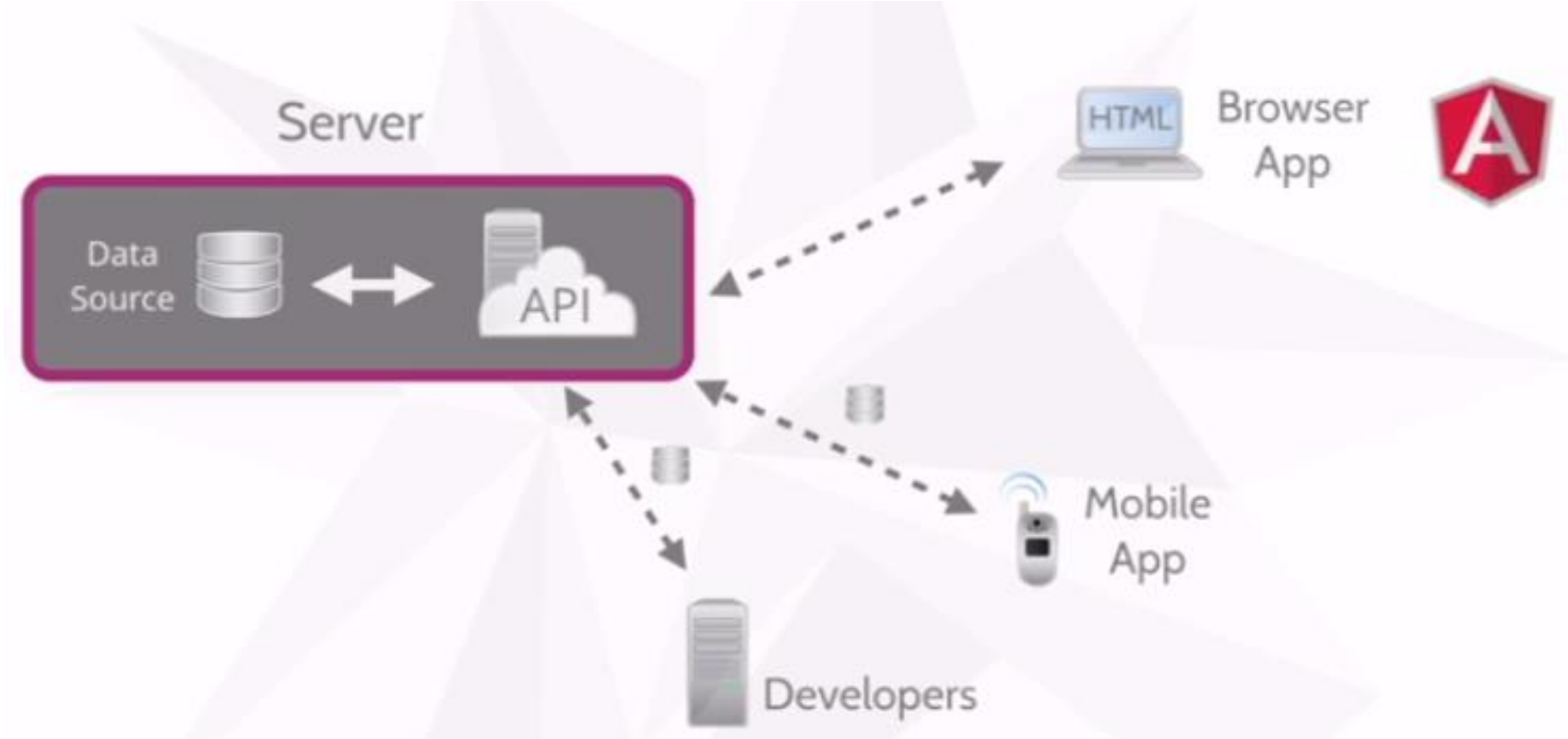
Web API

- Là các API được truy xuất thông qua giao thức HTTP:
  - Yêu cầu/Đáp ứng được thực hiện bằng giao thức HTTP
  - Có thể được xây dựng bằng nhiều công nghệ khác nhau như: Java, .NET, PHP,…
- Phù hợp cho giao tiếp máy tính – máy tính
  (vs. các website: giao tiếp người – máy tính)
- Yêu cầu/Đáp ứng ở định dạng text và dưới dạng máy tính có thể phân tích và "hiểu" được như JSON, XML…
  (vs. HTML chỉ dùng để định dạng dữ liệu)
  - **Đầu vào**: URI [+JSON/XML/…], **đầu ra**: JSON/XML/…

Web API

- Demo Twitter REST API

Ứng dụng hướng API



Căn bản về REST

REST Basics

REST là gì?

- REST = Representational State Transfer
  - Một mô hình kiến trúc phần mềm, dùng trong thiết kế các ứng dụng mạng
  - Là mô hình thiết kế phổ biến nhất cho các Web API
- Ràng buộc trong kiến trúc REST:
  - Client/Server, stateless, cache, uniform interface, layered system, code-on-demand (optional)
- Các khái niệm cơ bản: Tài nguyên (resource), trạng thái đại diện (representation) và thao tác trên tài nguyên (manipulation of resources)

Tài nguyên

- Là một khái niệm trừu tượng:
  - Là bất kỳ thông tin, khái niệm nào có thể được đặt tên
  - Là các danh từ
- Xác định bởi các URI (Uniform Resource Identifier)
  - Nhiều URI có thể tham chiếu cùng 1 tài nguyên
- Có 2 dạng tài nguyên:
  - Dạng tập hợp (collection resource)
    - Ví dụ: /lectures
  - Dạng thể hiện (instance resource)
    - Ví dụ: /lectures/001533

Trạng thái đại diện

- Là dữ liệu hay trạng thái miêu tả/đại diện cho tài nguyên
  - Thông tin trao đổi giữa client và server
- Thông thường ở dạng JSON hoặc XML
- Ví dụ:
  - Tài nguyên: person
  - Trạng thái đại diện: { “name”: “Perter Jackson”,
    “address”: “123 Ada Street, PN”,
    “phone”: “(+64) 123456”

Các thao tác trên tài nguyên

- GET (Idempotent) = đọc (read)
- DELETE (Idempotent) = xóa (delete)
- POST = tạo mới hoặc cập nhật (create hoặc update)
- PUT (Idempotent) = tạo mới hoặc cập nhật (create hoặc update)

- Mỗi tài nguyên nên hỗ trợ trả lời cho tất cả các thao tác này (nếu không hỗ trợ thì trả về thông báo lỗi)

Tạo tài nguyên với PUT

- Tạo tài nguyên: cung cấp ID và thông tin của tài nguyên mới
  - PUT <new resource URI>
    {
      . . . //các thuộc tính của tài nguyên mới
    }
- Ví dụ, tạo một lecturer mới:
  - PUT /lectures/clientSpecifiedID
    {
      //các thuộc tính lecturer mới
    }

Cập nhật tài nguyên với PUT

- Cập nhật dữ liệu, trạng thái của tài nguyên:
  - PUT <existing resource URI>
    {
      . . . //thuộc tính của tài nguyên hiện có
    }
- Ví dụ, cập nhật thông tin của một lecturer
  - PUT /lecturers/existingID
    {
      “name”: “Bùi Võ Quốc Bảo”,
      “dept”: “IT”
    }

Tạo tài nguyên mới với POST

- Tạo tài nguyên mới:
  - POST <new resource ID>
    {
      . . . //các thuộc tính cho tài nguyên mới
    }
- Ví dụ, tạo 1 lecturer mới (ID được cấp tự động):
  - POST /lecturers
    {
      “name”: “Bùi Võ Quốc Bảo”
    }

Cập nhật tài nguyên với POST

- Cập nhật tài nguyên đã có sẵn:
  - POST <existing resource ID>
    {
      . . . //các thuộc tính cho tài nguyên sẵn có
    }
- Ví dụ, cập nhật thông tin của 1 lecturer:
  - POST /lecturers/003025
    {
      “name”: “Nguyen Van A”
    }

Nguyên tắc thiết kế REST API

- Tài nguyên phải là danh từ
- Tất cả tài nguyên nên hỗ trợ cả 4 thao tác GET, PUT, POST và DELETE
- Tất cả các thông tin trạng thái phải được giữ phía client (VD: nếu client cần thực hiện 3 bước thì client phải ghi nhớ là nó đang thực hiện tới bước nào)
- Không yêu cầu client phải tự xây dựng các URI mà server cần gửi URI của các tài nguyên trong câu trả lời về cho client (HATEOAS)

HATEOAS

- HATEOAS: Hypermedia as the Engine of Application State

```
Request: GET /api/v1/cars/711
Response:
{
  "id": 711,
  "manufacturer": "bmw",
  "model": "X5",
  "seats": 5,
  "drivers": [{
  "id": "23",
  "name": "Stefan Jauker",
  "links": [{"rel": "self",
             "href": "/api/v1/drivers/23"}]}]
}
```

Chứng thực người dùng REST API

REST API Authentication

Chứng thực dựa trên thẻ bài (Token)

- Người dùng được hệ thống cấp phát cho một API token
  - Token: một chuỗi được sinh ra ngẫu nhiên
- API token sẽ được đính kèm trong mỗi yêu cầu đến server (thông qua giao thức HTTPS)
  - curl -X GET https://127.0.0.1/api/example
    -H 'X-Api-Key: 9944b09199c62bcf9418ad846dd0e4bety89eff'
- Server sẽ kiểm tra tính hợp lệ của token trong yêu cầu

Chứng thực dựa trên HMAC

- HMAC: Hash-based message authentication code
- Sử dụng một cặp giá trị hash: private hash và public hash
  - Public hash dùng để định danh người dùng, có thể công khai
  - Private hash cần được giữ bí mật (chỉ có người dùng và server biết)
- Private hash được dùng để tạo hash cho nội dung của yêu cầu sẽ gửi đến server $\Rightarrow$ content hash
  - Content hash + public hash + yêu cầu sẽ được gửi đến cho server

Chứng thực dựa trên HMAC

- Server dựa vào public hash để truy vấn private hash tương ứng
- Server dùng private hash tìm được để tạo lại giá trị hash từ nội dung yêu cầu
- So sánh kết quả với content hash nhận được trong yêu cầu, nếu khớp $\Rightarrow$ yêu cầu hợp lệ

Chứng thực dựa trên HMAC

- Tạo private hash và public hash:
  - Private hash và public hash được tạo không nên dựa trên bất kỳ thông tin nào liên quan đến người dùng
  - Ví dụ:
    $hash = hash('sha256', openssl_random_pseudo_bytes(32));
    Hoặc:
    $hash = hash('sha256', mt_rand());

Chứng thực dựa trên HMAC – Client

```php
//Client
<?php
$publicHash = '3441df0babc2a2dda551d7cd39fb235bc4e09cd1e4556bf261bb...';
$privateHash = 'e249c439ed7697df2a4b045d97d4b9b7e1854c3ff8dd668c779...';
$content = json_encode(array('test' => 'content'));

$hash = hash_hmac('sha256', $content, $privateHash);
$headers = array('X-Public: '.$publicHash,
                 'X-Hash: '.$hash);

$ch = curl_init('http://test.localhost:8080/api-test/');
curl_setopt($ch,CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch,CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch,CURLOPT_POSTFIELDS, $content);

$result = curl_exec($ch);
curl_close($ch);
echo "RESULT\n======\n".print_r($result, true)."\n\n";
?>
```

Chứng thực dựa trên HMAC – Server

```php
<?php
require_once 'vendor/autoload.php';
$app = new \Slim\App();
$app->post('/', function($request, $response) {
  $publicHash = $request->getHeaderLine('X-Public');
  $contentHash = $request->getHeaderLine('X-Hash');
  $privateHash = 'e249c439ed7697df2a4b045d97d4b9b7e1854c3ff8dd668c779...';

  $content = $request->getBody();
  $hash = hash_hmac('sha256', $content, $privateHash);

  if ($hash == $contentHash){
    echo "match!\n";
  }
});
?>
```

Cài đặt và tiêu thụ REST API

Creating and Using a REST API

Cài đặt REST API

- Đặt giá trị thích hợp cho response header “Content-Type”:
  - header('Content-Type: application/json');
- PHP array/object → Json string: json_encode($var)
  - $arr = ['a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5];
  - echo json_encode($arr); // {"a":1,"b":2,"c":3,"d":4,"e":5}
- Json string → PHP array/object: json_decode($var)
  - $json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
  - $obj = json_decode($json); // $obj is a stdClass object
  - $arr = json_decode($json, true); // $arr is an associative array

Định dạng trả lời theo đặc tả JSend

- Thành công (2xx): yêu cầu được xử lý thành công

```
{
  "status" : "success",
  "data" : {
  "post" : { "id" : 1,
  "title" : "A blog post",
  "body" : "Some useful content" }}
}

{
  "status" : "success",
  "data" : {
}
```

Định dạng trả lời theo đặc tả JSend

- Thất bại (4xx): yêu cầu không hợp lệ (lỗi client)

```
{
  "status" : "failed",
  "data" : {"title" : "A title is required"}
}
```

- Thất bại (5xx): lỗi trong quá trình xử lý y/c (lỗi server)

```
{
  "status" : "error",
  "message" : "Unable to communicate with database"
}
```

Làm việc với json trong Javascript

- JS object → Json string: JSON.stringify(obj)
  - `var j={"name":"binchen"};`
  - `JSON.stringify(j); // '{"name":"binchen"}'`
- Json string → JS object: JSON.parse(json)
  - `var json = '{"result":true,"count":1}';`
  - `var obj = JSON.parse(json);`

Tiêu thụ dữ liệu JSON với AJAX (jQuery)

- Tiêu thụ json từ server gửi về:

```
$.ajax({
  type: 'GET',
  url: '/api/contacts',
  data: {name:'Bao'},
  dataType:'json',
  success: function (res) {
    // jQuery tự chuyển đổi dữ liệu
    // json từ server sang js object.
    // res: JS object
  }
});
```

Tiêu thụ dữ liệu JSON với AJAX (jQuery)

- Gửi dữ liệu dạng json cho server:

```
var contact = {name:'Bao',phone:'1234567890'};
$.ajax({
  type: 'POST',
  url: '/api/contacts',
  data: JSON.stringify(contact),
  contentType:'application/json',
  success: function (res) {
    alert(res)
  }
});
```

Question?

CT275 – CÔNG NGHỆ WEB