AJAX

Nội dung

- Ajax là gì
- Công nghệ
- Hoạt động
- Lợi ích/ bất lợi

Lịch sử

- Internet Explorer giới thiệu khái niệm Iframe năm 1996
- Năm 1998, Microsoft giới thiệu Microsoft's Remote Scripting
- Năm 1999 Microsoft đưa ra XMLHttpRequest trong IE5
- AJAX được Jess James Garret đưa ra năm 2005 và được Google ứng dụng ngay sau đó.
- AJAX được W3C công bố vào năm 2006

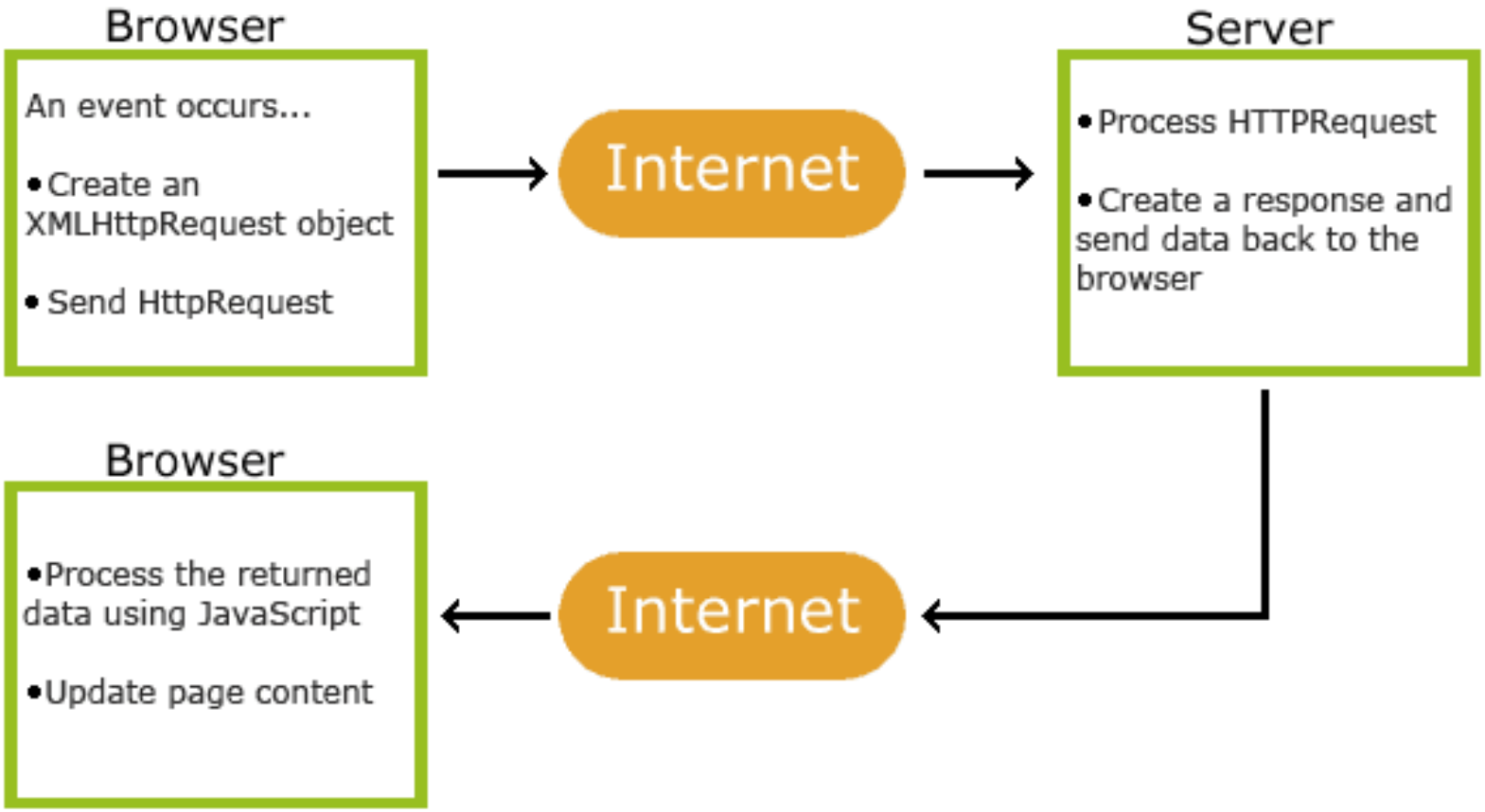
AJAX là gì

- AJAX – Asynchronous JavaScript and XML
- Cập nhật một phần của trang web mà không cần tải lại trang
- Không là ngôn ngữ hay công nghệ độc lập, phối hợp cùng các công nghệ có sẵn
- Công nghệ dành cho phía client
- Độc lập với phần mềm máy chủ web

Công nghệ

- HTML, XHTML, CSS
- Document Object Model (DOM) cho hiển thị dữ liệu động
- XML, JSON: mang dữ liệu tới/từ server
- XMLHttpRequest – đọc/ gửi dữ liệu giữa client và server bất đồng bộ
- JavaScript kết hợp chúng với nhau

Hoạt động



- Nguồn ảnh: W3School

- Một sự kiện xảy ra trên trang web (load, click button,…)
- XMLHttpRequest object được tạo bởi JS
- XMLHttpRequest object gửi yêu cầu đến server
- Server xử lý yêu cầu
- Server gửi đáp ứng
- Đáp ứng được JS đọc
- JS cập nhật lại trang

- Người dùng gửi request tới XMLHttpRequest
- HttpRequest được gửi tới server bởi XMLHttpRequest
- Server tương tác với server (qua ASP.NET, PHP,…)
- Truy vấn dữ liệu
- Server gửi dữ liệu (XML, JSON) tới XMLHttpRequest callback function
- HTML, CSS hiện thị dữ liệu lên trình duyệt

Hiểu về XMLHttpRequest

- Một đối tượng của XMLHttpRequest được dung cho truyền thông bất đồng bộ giữa client và server
  - Gửi dữ liệu từ client
  - Nhận dữ liệu từ server
  - Cập nhật webpage mà không cần tải trang
- Mọi trình duyệt hiện nay đều hỗ trợ XMLHttpRequest

Thuộc tính của XMLHttpRequest

Thuộc tính	Miêu tả
onReadyStateChange	Được gọi khi thuộc tính ReadyState có thay đổi. Nó không được dùng với các yêu cầu đồng bộ
readyState	Miêu tả trạng thái của request, có giá trị từ 0 đến 4 0 UNOPEN open() không được gọi 1 OPENED open được gọi nhưng send() không được gọi 2 HEADERS_RECEIVED send() được gọi và headers và trạng thái được sẵn sàng 3 LOADING Downloading data, responseText giữ dữ liệu 4. DONE hành xử hoàn thành
responseText	Trả về đáp ứng là text
responseXML	trả về đáp ứng là XML

Các phương thức của XMLHttpRequest

Phương thức	Miêu tả
void open(method, URL)	Mở một yêu cầu dùng get, post và url
void open(method, URL, async)	Giống trên, xác định đùng bất đồng bộ hay không
void open(method, URL, async, username, password)	Giống ở trên, thêm tên người dùng và mật khẩu
void send()	Gửi một yêu cầu get
void send(string)	Gửi một yêu cầu post
setRequestHeader(header, value)	Thêm một header yêu cầu

XMLHttpRequest

- Tạo một XMLHttpRequest object:

Variale=new XMLHttpRequest();

- Gửi một yêu cầu đến server, dùng open(), send():

xhttp.open(“GET”, “ajax_info.txt”, true);

xhttp.send();

- Sau khi tạo XMLHttpRequest object:

```
xmlhttp.onreadystatechange=function(){
        //xử lý phản hồi của máy chủ
}
```

Lợi ích

- Trang có thể làm tươi động
- Đáp ứng người dùng nhanh hơn
- Tải nhanh hơn do dung lượng tải ít hơn
- Giảm băng thông

Bất lợi

- Người dùng khó đánh dấu trạng thái của trang web
- Phụ thuộc vào JavaScript
- Khó được tìm kiếm
- Nút Back và refresh có thể bị vô hiệu

Ví dụ

- https://www.w3schools.com/xml/tryit.asp?filename=try_dom_xmlhttprequest_ie

Jquery và Ajax

- $(selector).load(URL, data, callback)
  - Selector: là thẻ HTML cần thay đổi dữ liệu
  - URL là địa chỉ
  - Gửi dữ liệu và nhận dữ liệu qua data
  - Hàm callback là những hành xử sau khi hoàn thành
  - Sự kiện, trạng thái: ajaxComplete(), ajaxStart(),…

Ví dụ

```
$(document).ready(function(){
        $("#txt").ajaxStart(function(){
                $("#wait").css("background-color", "red");
        });
        $("#wait").ajaxComplete(function(){
                $("#wait").css("background-color", "green");
        });
        $("button").click(function(){
                $("#txt").load("test.htm");
        });
});
```

Ajax object

- Url
- HttpMethod: GET, POST, PUT, DELETE,…
- InsertionMode: cách xử lý dữ liệu đã nhận
  - InsertAfter, InsertBefore, Replace
- UpdateTargetId- thẻ HTML được thay đổi nội dung
- LoadingElementId: show/ hide “loading…” khi đang tải
- Sự kiện (JS function)
  - OnSuccess, OnFailure, OnBegin, OnComplete

```
$.ajax({
        type: 'GET’/’POST’/’PUT’/’DETELE’
        url: ,
        dataType: 'json’/’xml’,
        success: function () {},
        error: function () {}
    });
```

$.ajax({ type: 'Loại gửi dữ liệu (POST hoặc GET)', url: 'URL Tiếp nhận, xử lý và gửi lại dữ liệu', data: { Các biến dữ liệu được gửi lên server.(ten_bien1:dữliệu,ten_bien2:dữliệu,...). Có thể sử dụng var data = $('form#ID_form').serialize(); để lấy toàn bộ dữ liệu từ 1 form có id là ID_form}, dataType: Kiểu dữ liệu trả về ('html','text','json','xml'), success: function(data) { Nội dung sẽ được thực thi sau khi nhận được dữ liệu từ server }, error: function() { Nội dung sẽ được thực thi khi có lỗi phát sinh } });

```
$.ajax({
        type: 'GET',
        url: 'https://localhost:44323/api/Products',
        dataType: 'json',
        success: function (data) {
            $.each(data, function (key, val) {
                var getURL = 'https://localhost:44323/api/Products/' +
                                    val.MaSP;
                str += '<span style="font-size:14px; cursor: pointer;
                                    onclick=ShowProdByID(\'' + getURL
                                    +'\'); return flase;">' + val.MaSP +
                                    '| </span>';
            });
            $('#ProdIDList').html(str);
        },
        error: function (xhr) {
            alert(xhr.responseText);
        }
    });
```

Ajax ActionLink Help

Định nghĩa một link (<a href=…) cho việc tải dữ liệu với ajax

```
@Ajax.ActionLink("Load Server Time", "ServerTime",null,
        new AjaxOptions{
                HttpMethod = "GET",
                UpdateTargetId = "timeDisplay",
                LoadingElementId = "timeDisplayLoading",
                InsertionMode = InsertionMode.Replace,
                OnBegin = "OnAjaxRequestBegin",
                OnFailure = "OnAjaxRequestFailure",
                OnSuccess = "OnAjaxRequestSuccess",
                OnComplete = "OnAjaxRequestComplete"
            }, new {@class="btn btn-primary"})
```

AJAX BeginForm Helper

- Một form submit yêu cầu Ajax và sinh ra partial view

```
@using(Ajax.BeginForm("Search", new AjaxOptions
{
    UpdateTargetId="results",
    InsertionMode=InsertionMode.Replace
}))
{
    <input type="text" name="query" />
    <input type="submit" />
}
<div id="result">@Html.Partial("_BookResult", Model)</div>
```

Ajax.BeginForm Helper

- Return một partial view tới helpers (không có layout)

```
public IHttpActionResult Search(string query)
        {
            var result = BooksData
                .GetAll()
                .AsQueryable()
                .Where(book => book.Title.Contains(query))
                .Select(BookViewModel.FromBook)
                .ToList();
            return this.PartialView("_BookResult", result);
        }
```

JSON

Nội dung

- JSON là gì?
- Cú pháp
- Kiểu dữ liệu
- Ứng dụng
-

JSON là gì

- JSON – JavaScript Object Notation là một định dạng chuyển đổi dữ liệu gọn nhẹ
- JSON là cấu trúc cho lưu trữ và chuyển đổi dữ liệu
- JSON dễ sử dụng hơn XML
- Dựa trên JavaScrip

JSON là gì

- JSON là định dạng text độc lập nhưng tương tự cới các ngôn ngữ lập trình quen thuộc: C, C++, Java, JavaScript, Python,…
- Kiểu file JSON là .json

Đặc điểm JSON

- Cú pháp đơn giản
- Kích thước nhẹ
- Dễ dàng tạo và thao tác
- Có thể được phân tích cú pháp trong JS sử dụng eval()
- Được hỗ trợ bởi JavaScript Framework
- Được hỗ trợ bởi hầu hết công nghệ back end
- Thường được sử dụng trong các ứng dụng Ajax

Điểm mạnh và yếu

Mạnh	Yếu
Phân tích cú pháp nhanh	Không hỗ trợ namespace, do vậy khả năng mở rộng kém
Hỗ trợ tốt mọi trình duyệt	Hỗ trợ công cụ phát triển hạn chế
Được hỗ trợ bởi nhiều ngôn ngữ	Không hỗ trợ cho định nghĩa ngữ pháp chuẩn
Định dạng ngắn gọn: phương pháp tiếp cận dựa trên cặp name/value	
Đọc dễ dàng	
Viết dễ dàng	

Cú pháp JSON

- Bắt đầu bởi “{“, kết thúc bởi “}”
- Hỗ trợ hai cấu trúc dữ liệu:
  - Các thuộc tính, sử dụng cặp string: value, ngăn cách bởi dấu phẩy
  - Dãy giá trị: dùng dấu [], các phần tử ngăn cách nhau bởi dấu phẩy

Ví dụ

- Ví dụ:

```
var employeeData = {
  "employee_id": 1234567,
  "name": "Jeff Fox",
  "hire_date": "1/1/2013",
  "location": "Norwalk, CT",
  "consultant": false
};
```

Ví dụ

- Mảng

```
var employeeData = {
  "employee_id": 1236937,
  "name": "Jeff Fox",
  "hire_date": "1/1/2013",
  "location": "Norwalk, CT",
  "consultant": false,
   "random_nums": [ 24,65,12,94 ]
};
```

Kiểu dữ liệu

- Number: real, integer, float
- String
- Boolean
- Null
- Object
- Array

Cú pháp

- Tên thuộc tính: Giá trị thuộc tính
  - "Tên thuộc tính": "Giá trị thuộc tính"
  - "Tên thuộc tính": Giá trị thuộc tính
- Name phải đặt trong dấu nháy kép
- Chuỗi trong nháy kép, số viết thông thường
- Boolean viết true, false thông thường

JSON dùng ở đâu?

- Truyền và lấy dữ liệu từ Server
- Thực hiện đồng bộ dữ liệu mà không yêu cầu tải lại trang
- Làm việc với kho dữ liệu
- Biên dịch và lưu form dữ liệu người dùng cho lưu trữ cục bộ

Json Object

- {“student”:{
  “firstName”: John,
  “age”: “20”
  }}

Document.write(“)

WEB API MVC

Nội dung

- HTTP
- API là gì
- Các loại API
- Web API

HTTP

- Hypertext Transfer Protocol (HTTP)
- URI (Uniform Resource Identifier)
  - URL (Universal Resource Locator)
  - URN (Universal Resource Name)

URIs cho định danh

- Các thành phần của một URI (Uniform Resource Identifier) bao gồm:
  - Tên cơ chế: định danh giao thức: FTP, HTTP, HTTPS, IRC
  - Phần thứ bậc (Hierarchical Part): có xu hướng giữ thứ bậc thông tin tự nhiên
  - Truy vấn: chứa thông tin định danh bổ sung trong tự nhiên và thường phân tách bởi dấu hỏi (?)
  - Fragment: cùng cấp hướng đến tài nguyên thứ cấp trong chính một định danh bằng Authority và Path và ngăn cách bởi dấu #

/api/task

/api/task/1234

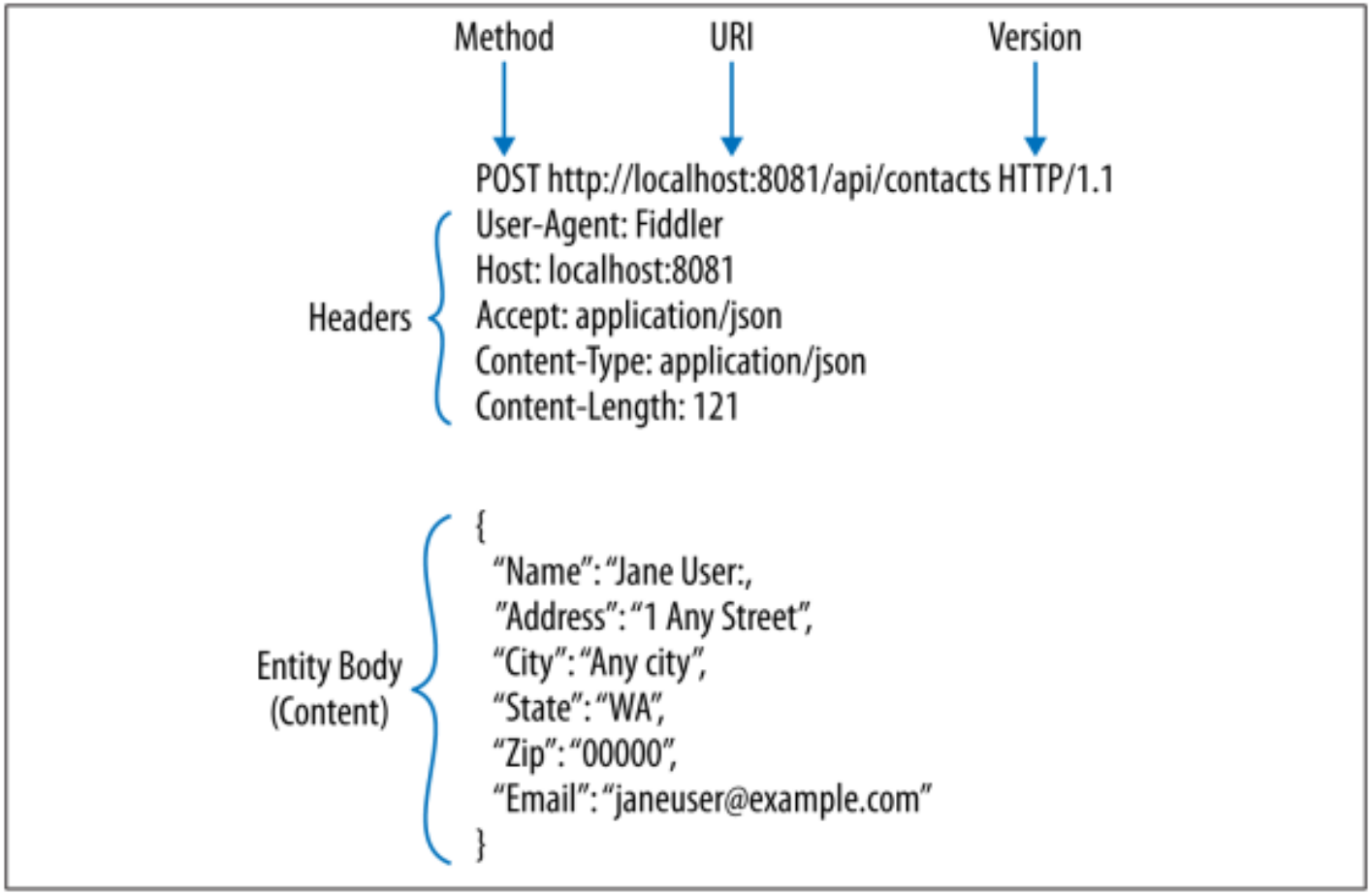
HTTP method (1)

- GET: lấy thông tin từ dữ liệu (status code 200 (OK))
- HEAD: giống GET song chỉ lấy tiêu đề, không lấy nội dung
- POST: yêu cầu server chấp nhận một thực thể vào dữ liệu. Server có thể tạo mới, nếu thành công sẽ trả về 201 (Created) hoặc 202 (Accepted) và location header. Nếu không tạo dữ liệu, sẽ trả về 200 (OK), 204 (No Content).
- PUT: yêu cầu server thay thế một trạng thái của dữ liệu tại một URI xác định với thực thể đi kèm. Nếu có máy chủ trả về 200 hoặc 204. Nếu không server có thể tạo mới, nếu được tạo sẽ trả về 201.

HTTP method (2)

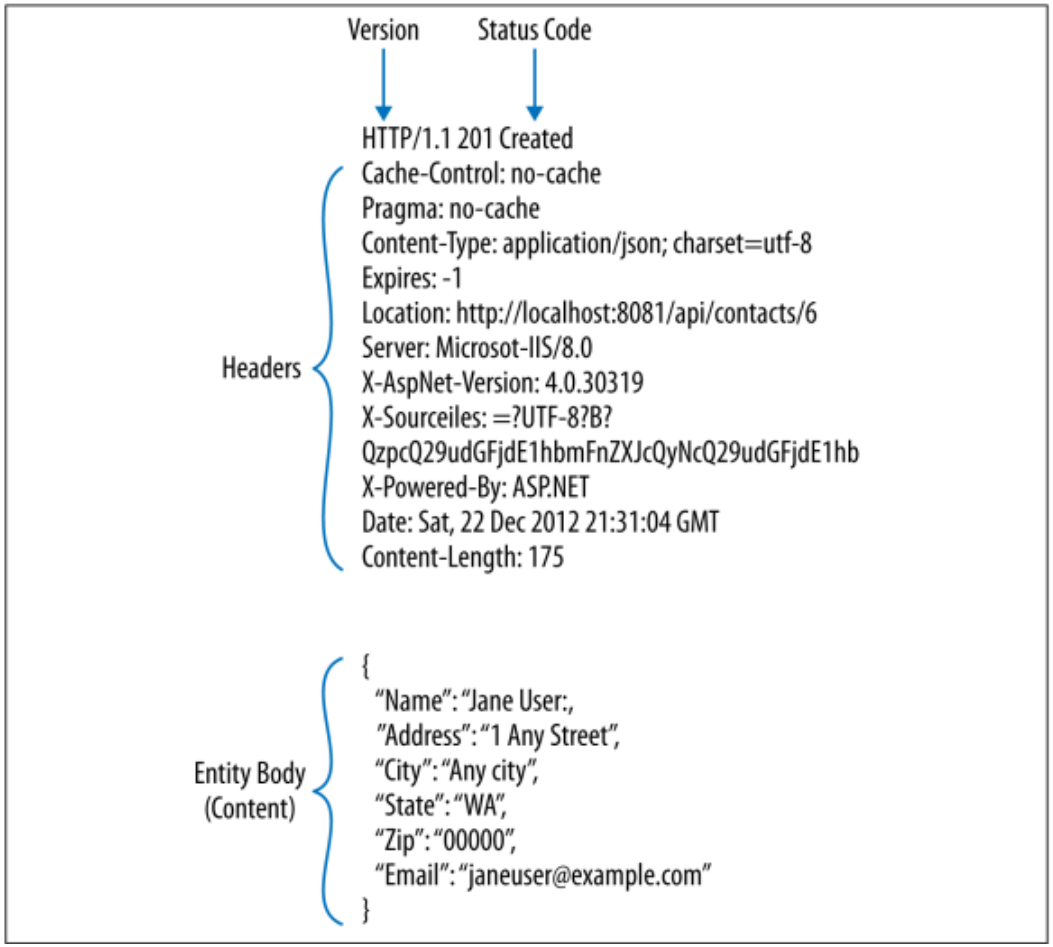
- DELETE: yêu cầu server xóa đi một thực thể trong một URI xác định. Server thực hiện ngay 200 code, nếu trì hoãn 202, 204.
- OPTIONS: yêu cầu server thông tin về khả năng. Thường trả về Allow header xác định các HTTP methods được hộ trợ.
- PATCH: yêu cầu server cập nhật một phần thực thể tại một URI chỉ định. Nếu dữ liệu tồn tàu, server có thể được cập nhật và trả về 200 hoặc 204.
- TRACE: yêu cầu server trả về yêu cầu được nhận. Server sẽ trả về toàn bộ thông điệp yêu cầu trong phần thân cùng với kiểu thông điệp, http.

HTTP Requets



Nguồn: Sách: Designing Evolvable Web APIs with ASP.NET

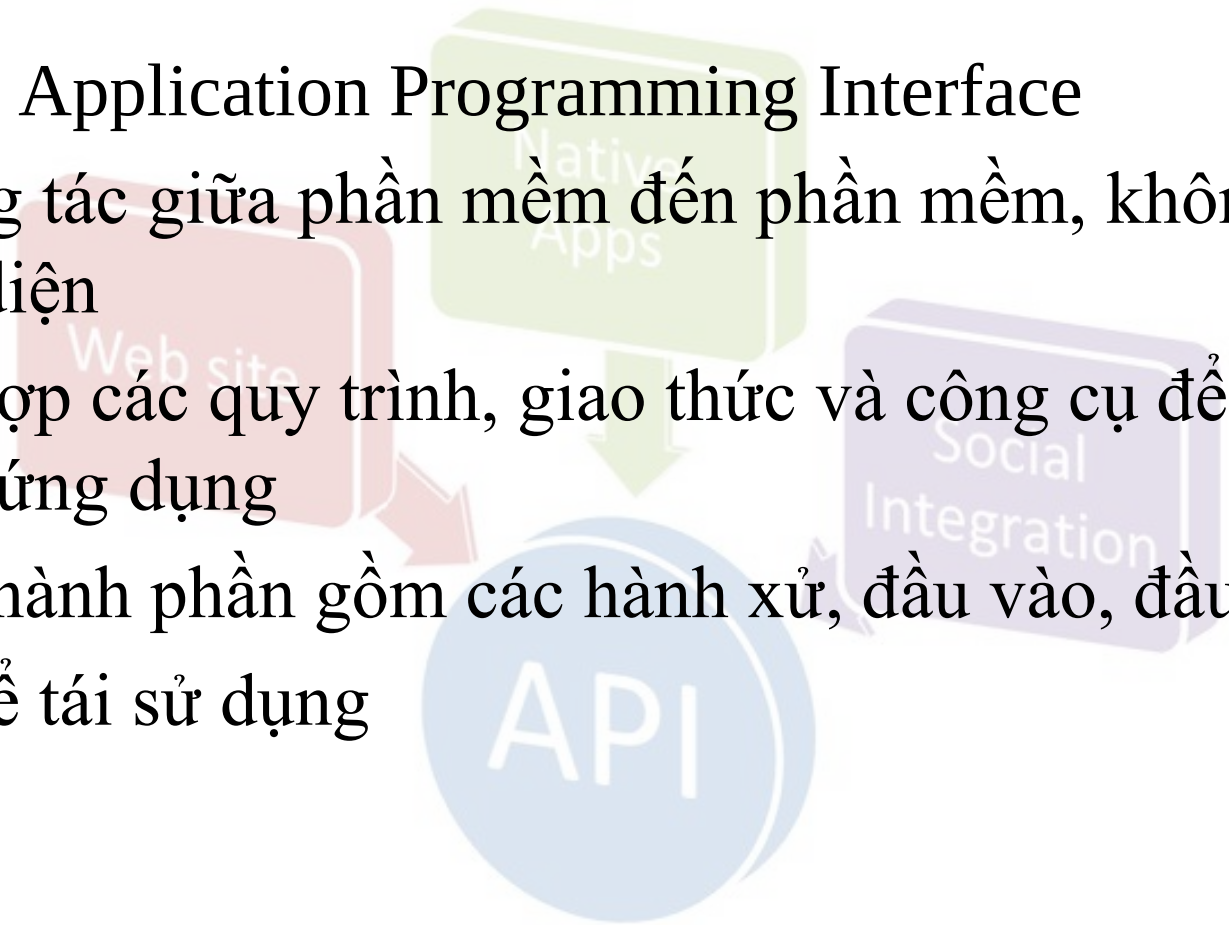
HTTP Respone



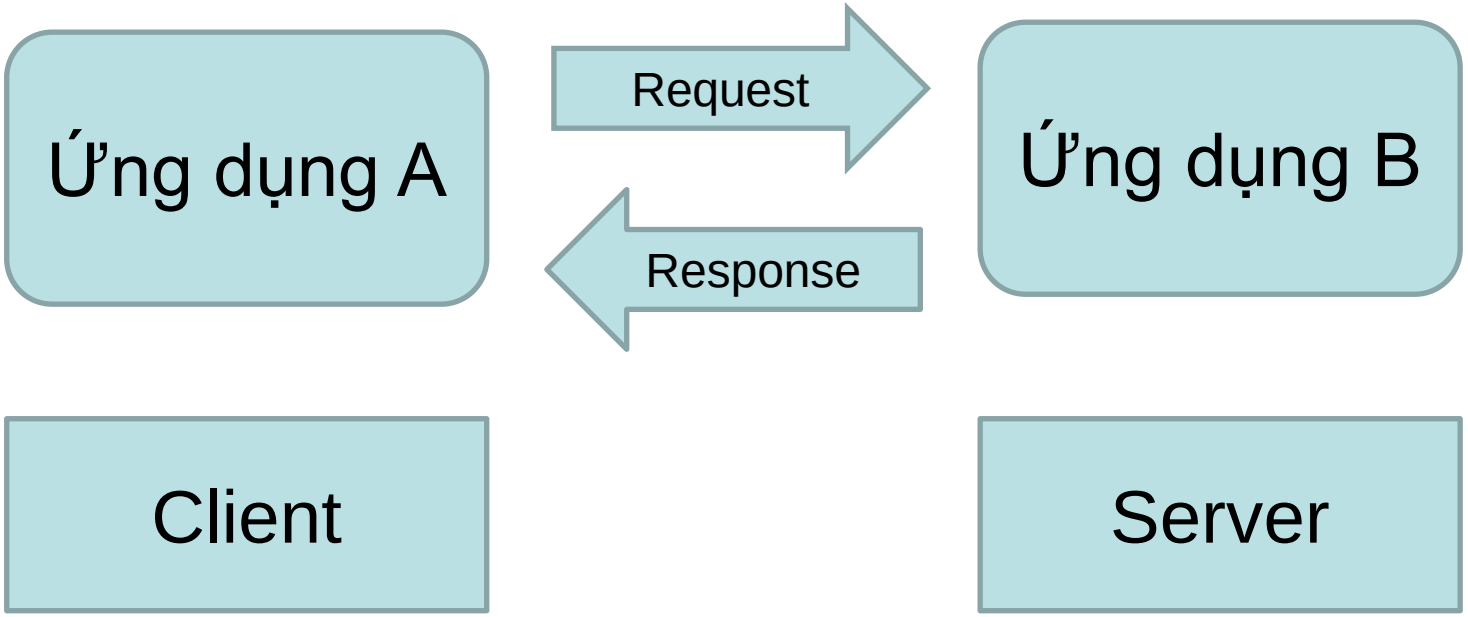
Nguồn: Sách: Designing Evolvable Web APIs with ASP.NET

API là gì

- API – Application Programming Interface
- Tương tác giữa phần mềm đến phần mềm, không có giao diện
- Tập hợp các quy trình, giao thức và công cụ để xây dựng ứng dụng
- Một thành phần gồm các hành xử, đầu vào, đầu ra.
- Có thể tái sử dụng



API



Các loại API

- Local API: là API truyền thống, cung cấp các ứng dụng cho ứng dụng (frontend/ GUI), yêu cầu dịch vụ hoặc ứng dụng từ backend như âm thanh hoặc dữ liệu từ DB
- Program API: dựa trên công nghệ RPC (Remote Procedure Call) thực hiện các chương trình từ xa từ một mấy chủ khác: SOA (Service Oriented Architecture) API là một Program API
- Web API: được biết đến như là Web Service, là một giao tiếp ứng dụng/ thiết bị thông qua WWW (trên HTTP)

Web APIs

- Web API là một giao diện
- Đầu ra thường được định dạng là XML hoặc Json
- Thường được thiết lập như một bộ các đáp ứng sử dụng giao thức HTTP

Web Services

- Có hai phương thức:
  - SOAP (Simple Object Access Protocol)
  - REST (Representational State Transfer)
- SOAP: giao thức cho việc trao đổi dữ liệu có cấu trúc trên nên tác phân tán sử dụng XML (chuẩn đầu ra là XML)
  - Tài chính
  - Viễn thông
  - Cổng thanh toán
- REST: là URL duy nhất là đại điện một số đối tượng, hỗ trợ nhiều định dạng: XML, JSON,…
  - Mạng xã hội
  - Web chat
  - Mobile

SOAP API

- Miêu tả cách chuẩn để tích hợp ứng dụng web, sử dụng XML, SOAP, WSDL, UDDI
  - XML (eXtensible Markup Language)
  - SOAP (Simple Object Access Protocol)
  - WSDL (Web Services Description Language)
  - UDDI (Universal Description, Discovery, and Integration)

REST API

- RESTful (Representational State Transfer)
- Sử dụng một HTTP với các phương thức GET, POST, PUT, DELETE, POST để chỉnh sửa đối tượng
- GET (Select): hiển thị tất các nhiệm vụ
- POST (Insert): tạo một nhiệm vụ mới
- GET (Select, Where): hiển thị một nhiệm vụ theo ID
- PUT (Update): cập nhật một nhiệm vụ theo ID
- DELETE (Delete): Xóa một nhiệm vụ theo ID

Khi nào dùng SOAP

- Đôi khi dễ triển khai
- Cứng nhắc: kiểm tra kiểu, tuân thủ hợp đồng (nhà cung cấp và người dung) phải đồng ý về hình thức trao đổi
- Nhiều công cụ phát triển
- Có thể sử dụng hầu hết các phương thức đẻ gửi yêu câu: SMTP (Simple Mail Transfer Protocol), JMS (Java Messaging Service)
- Xử lý không đồng bộ: bảo đảm mức độ tin cậy và bảo mât cho các hoạt động này
- Hoạt động trạng thái: hỗ trợ các thông tin quản lý trang thái thông tin và đối thoại (bảo mật, giao dịch, diều phối,…)

Khi nào sử dụng REST

- Nhẹ: các yêu cầu và đáp ứng ngắn
- Có thể đọc được kết quả: mềm dẻo, đơn giản, URIs cho định danh
- Dễ làm
- Không trạng thái: hành xử không trạng thái cho CRUD (Creat, Read, Update, Delete)
- Lưu bộ nhớ đệm: thông tin có thể được lưu bộ nhớ đệm do không có trạng thái