Lập trình AJAX

VỚI CÁC NGÔN NGỮ PHÍA SERVER

Bạn từng mơ ước xây dựng được một ứng dụng Web với cách thức hoạt động giống như một ứng dụng để bàn? Và bạn vẫn nghĩ rằng: dù cho sự phát triển của Web có rộng lớn đến thế nào chăng nữa, người dùng vẫn tiếp tục phải lặp đi lặp lại những thao tác kích chuột, đợi chờ cả trang web tải về, rồi lại kích chuột, lại đợi và cứ thế, cứ thế? Đấy là lí do khiến Ajax được khai sinh!

Với Ajax, bạn liên lạc với máy chủ web (web server) một cách "ngầm định", thu thập thông tin cần lấy và rồi hiển thị nó ngay lập tức trên website - không cần chờ đợi, cũng chẳng phải load nhiều lần. Thật tuyệt, với kết nối Internet ngày càng nhanh, ứng dụng web của bạn đã có cơ hội để trở thành một chương trình máy tình để bàn gần hơn bao giờ hết!

Đấy chính là tương lai của lập trình web - nơi những website sẽ chẳng khác là bao so với các ứng dụng được cài đặt trực tiếp trên máy tính cá nhân và đó cũng chính là chủ đề mà bài viết này đề cập với bạn!

I. Viễn cảnh tươi sáng dành cho những nhà phát triển

Trong cuốn sách của mình giới thiệu về Ajax, Jesse James Garrett - người được coi là "cha đẻ" của thuật ngữ này - đã chỉ ra rằng Ajax sẽ tác động mạnh mẽ đên thế giới Web như thế nào. Ông nhấn mạnh, mặc dù đã có rất nhiều sáng kiến được nêu ra nhằm xây dựng các ứng dụng trực tuyến, những nhà phát triển Web vẫn nhận thấy rằng việc biến ứng dụng web trở thành một phần mềm để bàn mạnh mẽ vẫn còn nằm ngoài tầm với của họ. Song, Ajax đã giúp thu hẹp khoảng cách đó!

Vậy bằng cách nào mà Ajax lại làm được điều này? Giải thích một cách đơn giản, AJAX được tạo thành từ cụm từ "*Asynchronous JavaScript and XML*" hay dịch nôm na là "XML và Javascript bất đồng bộ". Chúng gồm một vài thành phần cơ bản sau:

- Một trình duyệt có khả năng diễn tả HTML và bảng biểu mẫu CSS

- Dữ liệu được chứa dưới định dạng XML và được nhận về từ phía máy chủ

- Đồng thời, thông tin gửi lên sẽ sử dụng đối tượng XMLHttpRequest

- Tất cả chúng được nối kết với nhau thông qua việc sử dụng JavaScript

Quá trình làm việc của Ajax có thể được hình dung như sau:

- Tại phía trình duyệt, bạn sẽ viết mã JavaScript để xử lí dữ liệu nhận được từ phía server khi có yêu cầu hiển thị từ phía người dùng

- Khi nhiều thông tin cần được cung cấp thêm cho server, JavaScript sẽ sử dụng một đối tượng đặc biệt của trình duyệt là *XMLHttpRequest* để truyền yêu cầu này một cách "thầm lặng" mà không làm cho trang web phải refresh. JavaScript tại trình duyệt sẽ không "treo cứng" bất kỳ tác vụ nào trong khi nó chờ đợi dữ liệu gửi trả về từ phía server và ngay lập tức thực thi khi nhận được chúng. Đó chính là lý do mà Jesse James Garrett sử dụng từ "Asynchronous - bất đồng bộ" để diễn tả về Ajax.

- Dữ liệu được trả về từ phía server sẽ có dạng XML (hoặc dạng thuần văn bản). Mã JavaScript lúc này sẽ đọc chúng và hiển thị lên ngay tức thì.

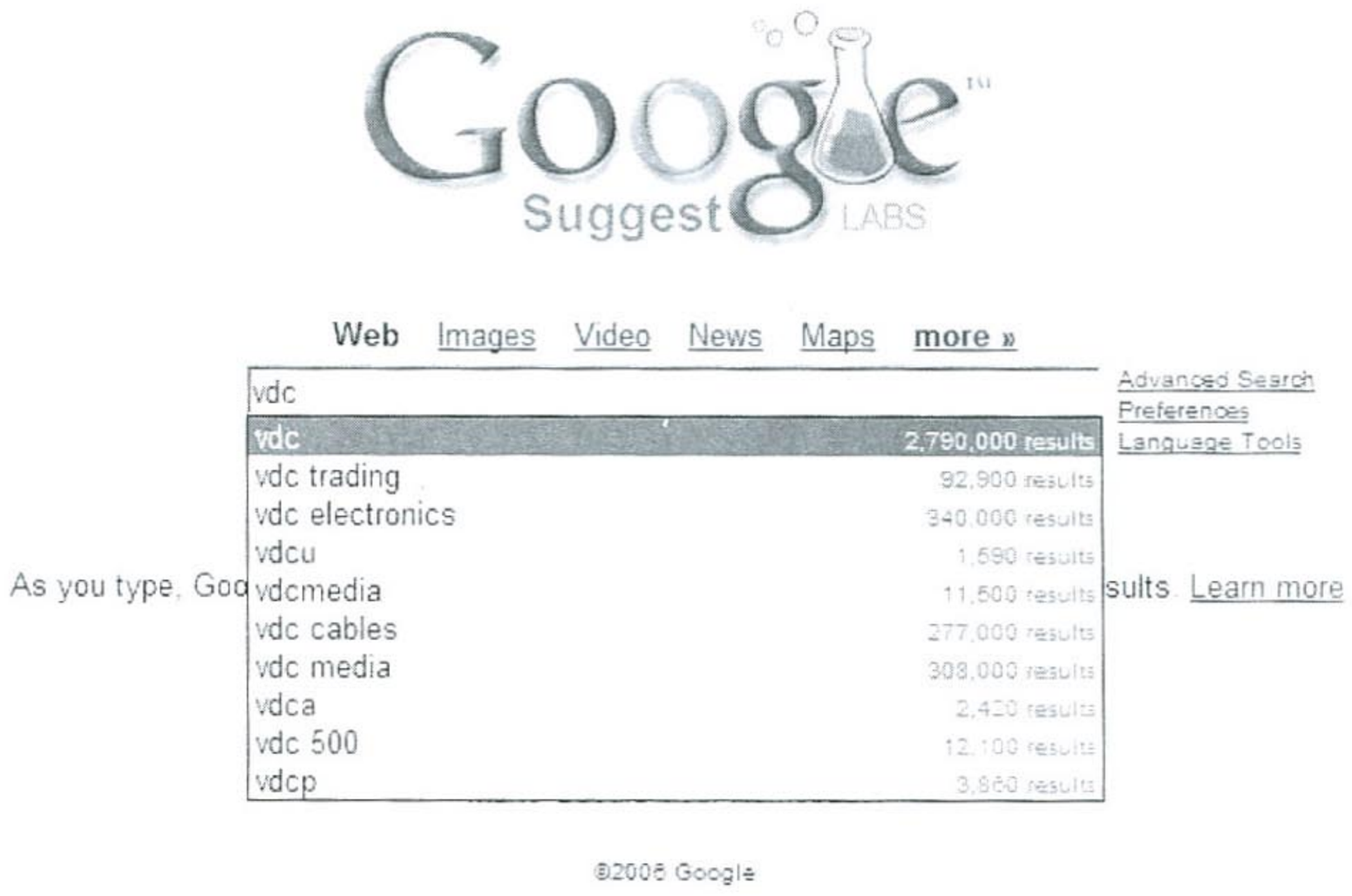
Nói ngắn gọn, Ajax làm việc dựa trên mã Javascript tại trình duyệt kết hợp với đối tượng *XMLHttpRequest* để liên lạc tới server mà không cần refresh lại trang, sau đó tiếp tục xử lí dữ liệu XML trả về từ server này.

Có lẽ cũng không cần phải nhắc lại quá nhiều về lịch sử hình thành của Ajax vì chắc hẳn bạn đọc còn nhớ một bài viết đã được đăng trước đây trên Tạp chí BCVT&CNTT cũng đã nêu khái quát về vấn đề này. Chỉ xin được nhắc lại rằng, Ajax không phải là một công nghệ mới mà nó đã được "manh nha" thành hình từ những năm 1997,1998 (trong sản

phẩm Outlook Web Access của Microsoft chẳng hạn). Tuy nhiên, cho đến tận năm 2005, khi "đại gia" Google vào cuộc cùng Ajax với những sản phẩm như *Google Suggest*, *Google Maps* và sự ra đời của thuật ngữ "Ajax" từ Jesse James Garrett, Ajax mới trở nên phổ biến. Và cũng kể từ đó, người ta nhận ra rằng: Phần mềm Web sẽ không còn cách xa phần mềm để bàn lâu hơn nữa.

Vậy, Ajax đã làm được những gì?

Có thể kể đến rất nhiều ứng dụng được xây dựng nên dựa trên Ajax. Nổi tiếng nhất có thể kể đến Google Maps - dịch vụ bản đồ trực tuyến miễn phí lớn nhất thế giới của Google. Chỉ cần vài cái kích chuột, kéo và thả, bạn đã có thể thấy được đến tận từng nóc nhà trong con phố mình đang sống dù bạn đang truy cập tại bất kỳ nơi đâu (http://maps.google.com). Hay như Google Suggest - dịch vụ tìm kiếm "đoán trước" từ khóa của Google. Để sử dụng dịch vụ này, bạn có thể truy cập tới địa chỉ http://www.google.com/webhp?complete=1&hl=en để thử nghiệm tính năng độc đáo đó. Chỉ cần gõ vào một từ khóa (hoặc một phần từ khóa), Google Suggest sẽ "đoán" biết được bạn định tìm kiếm dạng thông tin nào và làm xuất hiện một danh sách xổ xuống chứa kết quả ứng với từng từ khóa đó ngay lập tức. Giả sử, bạn gõ từ "VDC" vào ô tìm kiếm, Google Suggest sẽ hiện ra 10 kết quả thường gặp nhất với từ khóa này như "vdc internet" hoặc "vdc media". Tương tự, nếu bạn chỉ gõ một phần của từ như "prog", Google Suggest sẽ "nối" thêm



Hình 1

vào giúp bạn thành những từ thường gặp như "programming" hay "programming language". Khi đó, bạn có thể chọn trong danh sách hiện ra bằng cách cuộn lên xuống phím mũi tên hoặc dùng chuột để kích vào. Sự "thần kỳ" mà Google Suggest có được chính là nhờ sử dụng thuật toán "tiên đoán" dạng kết quả mà người dùng mong muốn gặp nhất, chứ không phải là những từ khóa mà bạn hay dùng nhất. ở đây, bạn sẽ nhận thấy rằng, điểm khác nhau giữa Google Suggest và một phần mềm cài đặt trên Desktop (như từ điển Lạc Việt chẳng hạn) dường như không còn giới hạn!

II. Chào mừng bạn đến thế giới Ajax

Bây giờ, chúng ta cùng xét một ví dụ thực tế, nhỏ thôi nhưng đủ để giúp bạn hình dung được phần nào về cách thức hoạt động của Ajax. Tuy nhiên, để có thể chạy được ứng dụng này, bạn cần có một Web server trước đã. Vì nội dung chính của bài viết đề cập đến PHP và Ajax, xin được đề cử với các bạn một bộ cài đặt "tất cả trong một": WAMP. Đây là một bộ cài đặt nhanh, nhẹ và hoàn toàn miễn phí. WAMP tương đương với: Web, Apache, MySQL và PHP nên khi cài đặt xong, trên máy của bạn sẽ có đầy đủ bộ 3 ứng dụng cần thiết để chạy được tất cả các ví dụ xuất hiện trong chủ đề này. Bạn có thể tải WAMP về tại địa chỉ sau: http://www.wampserver.com/ (dung lượng file Setup xấp xỉ 19MB). Quá trình cài đặt WAMP hết sức dễ dàng, bạn chỉ cần làm theo các bước mặc định sẵn của chương trình là có thể cài đặt hoàn toàn thành công. Chỉ xin lưu ý nhỏ với bạn rằng, web server Apache sau khi được cài đặt xong sẽ sử dụng cổng 80 để liên lạc. Vì thế, nếu trên máy của bạn có một ứng dụng nào khác đã chiếm dụng cổng 80 thì bạn có hai cách để khắc phục: Một là gỡ bỏ các ứng dụng chiếm cổng 80 (như IIS chẳng hạn), hai là đổi cổng lắng nghe mặc định của Apache sang một cổng tự do

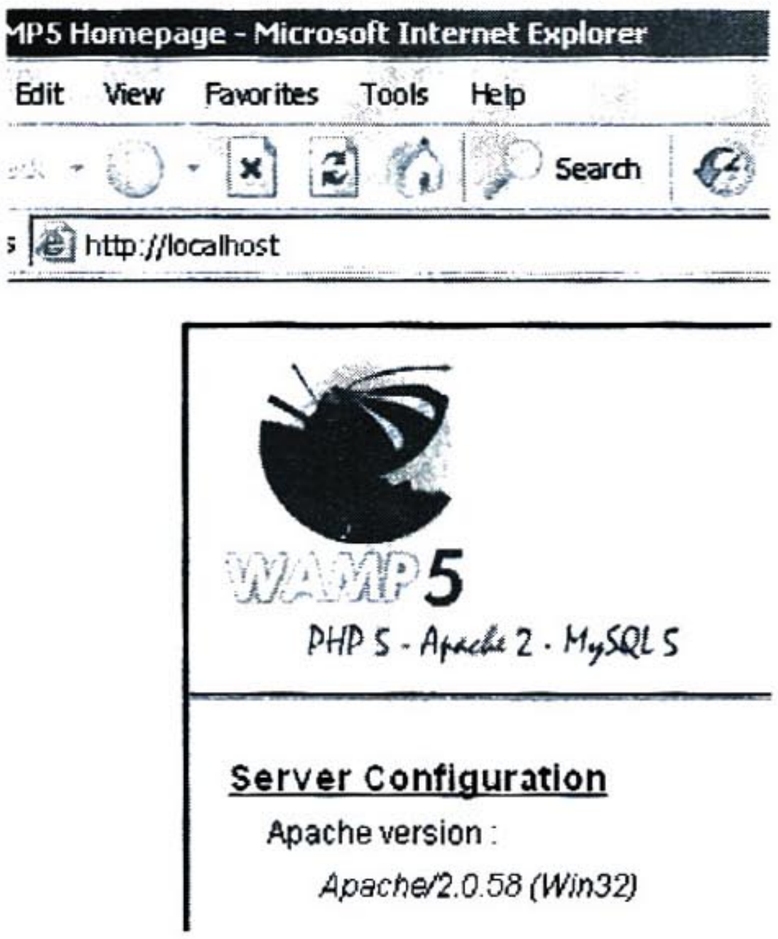
(chưa bị chiếm dụng) khác. Cách thức chuyển đổi hết sức giản đơn, bạn chỉ cần kích chuột trái vào biểu tượng WAMP (trên khay hệ thống), một menu sẽ hiện ra ngay lập tức, bạn chọn thư mục *Config Files* rồi trỏ đến *httpd.conf* để mở file này ra. Sau đó, dò tìm đến 2 dòng lệnh sau đây:

Listen 80

ServerName localhost:80

Rồi thay đổi số 80 bằng một cổng tự do bất kỳ (có thể là 8080 hoặc 8888, tùy bạn). Lưu file cấu hình này lại rồi restart server Apache theo cách sau: Kích chuột trái vào biểu tượng WAMP, chọn câu lệnh *Restart All Services* rồi đợi một lúc cho đến khi biểu tượng WAMP có màu trắng hoàn toàn là thành công! Bạn nên ghi nhớ kỹ điều này, mỗi khi thay đổi bất kỳ một thuộc tính nào trong các file cấu hình, hãy luôn restart server để mọi thay đổi có hiệu lực ngay lập tức. Nếu có một vài ứng dụng của WAMP không khởi động thành công, biểu tượng WAMP lúc đó sẽ có dạng như Hình 2. Thông thường, lỗi này là do đã có một ứng dụng chiếm mất cổng 80 của Apache, bạn chỉ cần bình tĩnh làm theo một trong 2 phương án khắc phục trên là được. Khi đó, server Apache đã có thể được truy cập thông qua địa chỉ mặc định sau: http://localhost (nếu bạn đã thay cổng mặc định 80, chỉ cần sửa lại địa chỉ truy cập cho chính xác là được). Nếu thành công, giao diện chào mừng của WAMP sẽ xuất hiện (Hình 2).

Cách thức dùng WAMP cũng không khó. Nếu bạn giữ nguyên những cấu hình sẵn có khi cài đặt WAMP, thư mục www của Apache (nơi mà bạn chứa các file web của mình) sẽ được đặt tại thư mục *C:\Wamp\www*. Bạn có thể tạo một thư mục mới dưới www để tiện quản lý hơn theo từng dự án sau này. Ở đây, bạn nên tạo một thư mục mới có tên là Ajax



Hình 2

và tất cả các ví dụ được đề cập đến trong bài viết này sẽ giả định rằng chúng được đặt dưới thư mục Ajax vừa tạo.

Nào, ta cùng nhau bắt tay vào ví dụ đầu tiên!

Trước hết, bạn hãy nhìn vào Hình 3 và nghe qua mô tả cách thức hoạt động của chương trình:

- Trên trang web (dạng HTML), có đặt một nút tên là *Click me*!

- Khi bạn kích vào nút này, một dòng chữ mới sẽ xuất hiện ngay lập tức bên dưới: "*Chào mừng các bạn đến với thế giới của AJAX*!" mà không hề có sự refresh lại cả trang.

Bạn nghĩ sao? Việc thay đổi một dòng chữ đơn giản như thế này chẳng có gì đặc biệt trong thế giới Web! Việc cần làm là đặt câu chào mừng ấy trong một đoạn JavaScript sẵn có ngay trong chính file HTML đó là xong? Nhưng, xin thưa với bạn rằng, lời chào mừng ấy không hề được đặt sẵn trong đó, nó được chứa trong một file văn bản có tên đầy đủ là *data.txt* (đặt trong thư mục Ajax của www) trên server Apache và sử dụng chính Ajax để "lôi" nó ra trình diện bạn! Vậy đoạn JavaScript đó đã làm như thế nào? Mời bạn cùng xem cụ thể đoạn mã trong file *index.html* (đặt cùng cấp với file *data.txt*):

index.html

```
<html><head><title>Welcome to Ajax</title>
<script language = "javascript">
var XMLHttpRequestObject = false;
if (window.XMLHttpRequest){
    XMLHttpRequestObject = new XMLHttpRequest();
}
else if(window.ActiveXObject){
    XMLHttpRequestObject = new ActiveXObject("Microsoft.XML-HTTP");
}
function getData(dataSource, divID){

if(XMLHttpRequestObject){
    var obj = document.getElementById(divID); //1

XMLHttpRequestObject.open("GET", dataSource);

XMLHttpRequestObject.onreadystatechange = function() {
```

```
if(XMLHttpRequestObject.readySt
ate == 4 &&

XMLHttpRequestObject.status ==
200){

obj.innerHTML =
XMLHttpRequestObject.response
Text;
                }
        }


XMLHttpRequestObject.send(null)
;
            }
    }
    </script></head>
    <body>
    <H1>Tạp chí Bưu chính viễn
thông</H1>
    <form>
    <input type = "button" value =
"Click me!"
onclick="getData('http://local-
host/Ajax/data.txt','targetDiv')">
    </form>
    <div id="targetDiv">
    <p>Nothing...</p>
    </div></body></html>
```

Riêng nội dung file *data.txt* chỉ đơn thuần gồm một dòng chữ duy nhất: "Chào mừng các bạn đến với thế giới của AJAX!". Cả 2 file *index.html* và *data.txt* nêu trên đều được đặt dưới thư mục Ajax trong www và có thể truy cập thẳng tới thông qua địa chỉ: http://localhost/Ajax/index.htm (nếu server Apache hoạt động tốt, bạn sẽ dễ dàng thấy được giao diện chính của trang web xuất hiện như Hình 3).

Do khuôn khổ có hạn, những đoạn mã HTML và javascript căn bản xin được phép bỏ qua không cần giải thích, bài viết chỉ tập trung hướng dẫn bạn đọc đi vào những đoạn mã mới, có áp dụng Ajax mà thôi. Trước hết, chúng ta hãy nhìn vào đoạn mã có chứa

Hình 3

thẻ bên trên. Hẳn những bạn đã từng lập trình web sẽ dễ dàng biết được tính năng của thẻ này. Và trong ví dụ mà ta xem xét thì chương trình đã sử dụng một thẻ có mã ID là *targetDiv* để hiển thị câu chào mừng sau khi bạn nhấp chuột vào nút *Kick Me!* Vậy, bằng cách nào bạn có thể tiếp cận đến nội dung của thẻ trong javascript? Câu trả lời là nhờ mã ID của chính nó! Với sự trợ giúp của đối tượng văn bản (tương trưng cho cả trang Web) và hàm quản lý *getElementById*, bạn có thể truy xuất đến một đối tượng bất kỳ trong trang miễn là nắm được ID của đối tượng đó. Ở đây, câu lệnh được comment (chú thích) bằng dấu //1 trong đoạn code kể trên có tác dụng tạo mới một object, đồng thời gán nó với thẻ có ID là *targerDiv*. Lúc này, để thể hiện nội dung cho thẻ , bạn sẽ sử dụng thuộc tính *innerHTML* để cập nhật vào. Xin nhớ rằng, ngoài ra, bạn cũng có thể xài thẻ với cách hoạt động y như thế. Điểm khác biệt duy nhất là luôn nằm trên một dòng riêng biệt, trong khi có thể được hiển thị chung nội dung với một dòng khác.

Chúng ta cùng tiếp tục đến với đối tượng *XMLHttpRequest*. Đây là một đối tượng đặc biệt và với mỗi trình duyệt, cách thức khai báo của chúng khác nhau đôi chút. Đoạn code ví dụ nêu trên là cách thức khai báo trong cả trình duyệt Internet Explorer lẫn Firefox (hoặc Netscape). Sau này, khi xây dựng các ứng dụng dùng Ajax, bạn nên lưu ý điều đó để chương trình của bạn có thể chạy tốt trên những nền tảng trình duyệt thông dụng nhất. Khi đã có được đối tượng *XMLHttpRequest*, bạn có thể sử dụng tất cả thuộc tính cũng như các hàm xây dựng sẵn cho đối tượng này. Trong ví dụ nêu trên, bạn đã dùng hàm *open()* để bắt đầu việc nhận dữ liệu từ server. Tuy nhiên, trong phần II này, chúng ta hãy tạm dừng ở đây để nhường việc đi sâu hơn vào Ajax cho phần III dưới đây:

III. Lập trình với Ajax

Đối tượng *XMLHttpRequest* như đã đề cập trước đây, là một đối tượng hết sức đặc biệt. Nó đặc biệt không chỉ trong cách khai báo ứng với từng trình duyệt, mà còn khác biệt ngay cả trong từng thuộc tính liên quan. Trong phần này, chúng ta sẽ cùng nhau tìm hiểu về những thuộc tính và các hàm xây dựng sẵn cho đối tượng *XMLHttpRequest* trên từng trình duyệt khác nhau.

Bây giờ, chúng ta tiếp tục tìm hiểu kỹ hơn về hàm *open()* của đối tượng *XMLHttpRequest*. Cú pháp lệnh đầy đủ khi sử dụng hàm *open()* được trình bày như dưới đây (xin bạn lưu ý rằng, các tham số trong dấu ngoặc vuông []

1.1 Thuộc tính của đối tượng *XMLHttpRequest* trên trình duyệt Internet Explorer

Tên thuộc tính	*Ý nghĩa*	*Truy xuất (đọc/viết)*
onreadystatechange	Lưu giữ dạng thức quản lý sự kiện được gọi khi giá trị của thuộc tính *readyState* thay đổi	Đọc/Viết
readyState	Lưu giữ trạng thái của truy vấn	Chỉ đọc
responseBody	Lưu nội dung truy vấn HTTP	Chỉ đọc
responseStream	Lưu chuỗi truy vấn nhị phân	Chỉ đọc
responseText	Lưu giữ truy vấn dạng String	Chỉ đọc
responseXML	Lưu giữ truy vấn dạng XML	Chỉ đọc
status	Lưu giữ mã trạng thái trả về từ truy vấn	Chỉ đọc
statusText	Lưu giữ nội dung HTTP trả về từ truy vấn	Chỉ đọc

1.2 Các hàm của đối tượng *XMLHttpRequest* trên trình duyệt Internet Explorer

Tên hàm	*Ý nghĩa*
abort	Hủy truy vấn HTTP
getAllResponseHeaders	Nhận về tất cả các HTTP header
getResponseHeader	Nhận về giá trị một HTTP header
open	Mở một truy vấn tới server
send	Gửi một truy vấn HTTP tới server
setRequestHeader	Đặt lại tên và giá trị cho một HTTP header

2.1 Thuộc tính của đối tượng *XMLHttpRequest* trên Mozilla, Firefoxvà Netscape

Tên thuộc tính	*Ý nghĩa*	*Truy xuất (đọc/viết)*
channel	Lưu giữ kênh được sử dụng cho truy vấn	Chỉ đọc
readyState	Lưu giữ trạng thái của truy vấn	Chỉ đọc
responseText	Lưu giữ truy vấn dạng String	Chỉ đọc
responseXML	Lưu giữ truy vấn dạng XML	Chỉ đọc
status	Lưu giữ mã trạng thái trả về từ truy vấn	Chỉ đọc
statusText	Lưu giữ nội dung HTTP trả về từ truy vấn	Chỉ đọc

2.2 Các hàm của đối tượng *XMLHttpRequest* trên Mozilla, Firefoxvà Netscape

Tên hàm	*Ý nghĩa*
abort	Hủy truy vấn HTTP
getAllResponseHeaders	Nhận về tất cả các HTTP header
getResponseHeader	Nhận về giá trị một HTTP header
openRequest	Mở một truy vấn tới se rver
overrideMimeType	Chồng toán tử dạng Mime mà server gửi trả về

là tùy chọn, không nhất thiết phải có):

open('method', 'URL' [, asyncFlag[, 'username' [, 'password']]])

Trong đó:

method: Tên phương pháp HTTP được sử dụng để mở kết nối. Có thể là GET, POST, PUT, HEAD hoặc PROFIND

URL: Địa chỉ truy vấn

asyncFlag: Một giá trị kiểu boolean tượng trưng cho quá trình bất đồng bộ. Mặc định của giá trị này là *true*

username: Tên người dùng

password: Mật khẩu

Thông thường, trong AJAX, ta hay dùng GET để nhận dữ liệu và POST để tải dữ liệu lên server. Vì thế, trong đoạn code ví dụ trên, GET được sử dụng trong *open* để lấy về dữ liệu chứa trong file *data.txt*. Bên cạnh đó, bạn cũng có thể sử dụng địa chỉ tương đối thay cho địa chỉ tuyệt đối cho giá trị URL tại ví dụ trên. Như thế, nếu file *data.txt* được đặt cùng thư mục với file *index.htm*, bạn có thể thay câu lệnh *open()* trong đoạn code mẫu trên bằng câu lệnh sau: *XMLHttpRequestObject.open("GET", 'data.txt');*

Bạn có thể hỏi rằng, giá trị *asyncFlag* có thể được dùng để để khống chế kết nối tới địa chỉ URL được phép đồng bộ, nhưng AJAX vốn bất đồng bộ (tức là nó không cần đợi kết nối hoàn thành và dữ liệu được tải về toàn bộ), vậy, chúng ta kiểm soát việc đấy như thế nào?

Một câu hỏi rất chính đáng!

Và xin được trả lời bạn rằng, chúng ta sẽ dùng thuộc tính *onreadystatechange* để thực hiện việc kiểm soát này. Trong đoạn code mẫu, có thể thấy rằng, chúng ta khởi tạo một hàm vô

danh dựa trên những thay đổi của thuộc tính *onreadystatechange*, *readyState* và *status*. Trong đó, thuộc tính *readyState* sẽ nói với bạn tình trạng load dữ liệu đang đến giai đoạn nào thông qua bốn trị số sau:

0: Chưa khởi tạo; 1: Đang tải; 2: Tải xong; 3: Có thể tương tác; 4: Hoàn thành

Ngoài ra, bạn cũng có thể dùng thuộc tính *status* để quản lý mã chuẩn HTTP trả về từ truy vấn và có thể được trình duyệt nhận ra. Một số mã chuẩn HTTP có thể kể đến (xin được giữ nguyên tên gốc tiếng Anh để giúp bạn tiện tra cứu sau này):

200: OK; 201: Created; 204: No content; 205: Reset content; 205: Partial Content; 400: Bad Request; 401: Unauthorized; 403: Forbidden; 404: Not Found; 408: Request Timeout; 500: Internal Server Error; 502: Bad Gateway; 503: Service Unavailable; 504: Gateway timeout; 505: HTTP Version Not Supported.

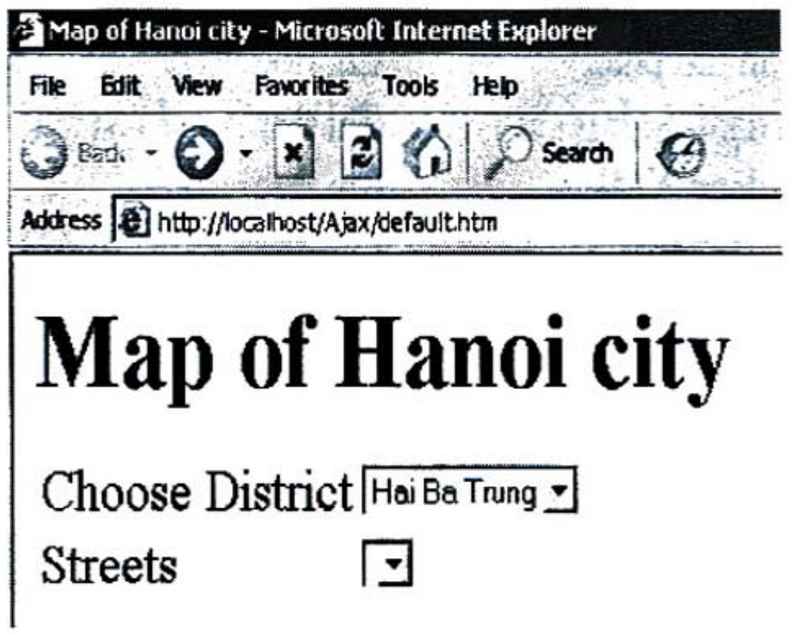
Do đó, trong đoạn code mẫu, để chắc chắn rằng mọi chuyện đều diễn ra tốt đẹp, chúng ta sẽ kiểm tra thông qua giá trị *readyState* bằng 4 và *status* bằng 200 trên mã JavaScript.

Lúc này, để nhận được giá trị trả về, bạn sẽ dùng đến một trong hai thuộc tính sau của đối tượng *XMLHttpRequest:*

- Nếu giá trị trả về là dạng text (thuần văn bản), bạn sẽ chỉ cần dùng đến thuộc tính *responseText*

- Nhưng nếu giả trị trả về được lưu trữ dưới định dạng XML, bạn phải dùng đến thuộc tính *responseXML*.

Như thế, trong ví dụ ta đang xét, *data.txt* đơn thuần là một file văn bản nên chỉ cần dùng *responseText* là đã lấy được giá trị để gán vào cho thẻ rồi. Việc cuối cùng cần làm lúc này



Hình 4

(sau khi đã có được tất cả đoạn mã cần thiết dành cho việc quản lý dữ liệu vào ra) chính là lúc ta *thật sự kết nối* đến server. Hàm *send()* được sử dụng cho mục đích đó, tham số *null* sẽ được truyền vào khi bạn dùng phương thức GET với server.

Như vậy, qua ví dụ đầu, bạn đã phần nào hiểu được cách thức lấy dữ liệu từ trên server bằng AJAX. Dạng dữ liệu này là thuần văn bản và thường được sử dụng khi bài toán hiển thị không yêu cầu mức tương tác cao giữa người dùng và cơ sở dữ liệu. Trong phần sau của bài viết, chúng ta sẽ cùng tìm hiểu sâu sát hơn khi áp dụng AJAX với các ngôn ngữ lập trình phía Server.

IV. AJAX và PHP

Mặc dù phần sau bài viết này chủ yếu đề cập đến việc áp dụng AJAX cùng ngôn ngữ lập trình PHP, nhưng trên thực tế, nguyên tắc làm việc của AJAX với bất kỳ ngôn ngữ lập trình phía server nào cũng rất tương đồng với nhau. Đó có thể là PHP, là JSP, là Perl hay thậm chí là Python đi chăng nữa thì nguyên tắc tương tác với AJAX của chúng đều không khác nhau là bao.

Như trên đã trình bày, bạn có thể thấy rằng, AJAX làm việc rất tốt với dạng dữ liệu thuần văn bản, nhưng trên hết, nó vẫn là JavaScript và XML bất đồng bộ. Và bây giờ là lúc chúng ta thực sự bắt tay vào Ajax + PHP cùng XML!

Trước hết, mời bạn cùng nhìn vào Hình 4 để xem ví dụ mà chúng ta sẽ xét đến trong phần này.

Hãy tạm gọi ứng dụng mà chúng ta vừa xem là "Bản đồ Hà Nội". Với ứng dụng này, người dùng chỉ cần chọn tên một quận trong mục "*Choose District*" là ngay lập tức, những con đường có trong quận này sẽ hiện ra trong mục "*Streets*" bên dưới mà không hề có sự refresh cả trang! Xin mô tả kỹ hơn với bạn, khi người dùng chọn tên một quận, hàm JavaScript trong trang sẽ tự động gửi tên quận này lên Server thông qua phương thức POST. Tại server, một file PHP ở đây sẽ xử lí thông tin mà bạn gửi lên, lọc nó ra và tạo ra một file XML chứa thông tin về toàn bộ con đường tại quận đấy rồi gửi trả lại client. Chính tại client, bạn sẽ phải viết một hàm JavaScirpt để xử lí thông tin về file XML này rồi hiển thị nó ra trong mục "Streets". Mời bạn xem toàn bộ đoạn code được sử dụng dưới đây:

default.htm

```
<html><head><title>Map of Hanoi city</title>
<script language =
```

```
"javascript">
    var XMLHttpRequestObject = false;
    if (window.XMLHttpRequest){
        XMLHttpRequestObject = new XMLHttpRequest();
    }
    else if(window.ActiveXObject){
        XMLHttpRequestObject = new ActiveXObject("Microsoft.XML-HTTP");
    }

    function GetDistrict(District){
        if(District != ""){
            SetDistrict("district="+District);
        }
    }

    var options;
    function SetDistrict(District){

if(XMLHttpRequestObject){

XMLHttpRequestObject.open('POST',"default.php", true); //1

XMLHttpRequestObject.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded'); //2

XMLHttpRequestObject.onreadystatechange = function() {
            if (XMLHttpRequestObject.readyState == 4 && XMLHttpRequestObject.status == 200){

var xmlDocument = XMLHttpRequestObject.responseXML; //3

options = xmlDocument.getElementsByTagName("option"); //4

listOptions();
            }
        }

XMLHttpRequestObject.send(District);
    }

    function listOptions(){
        var loopIndex;
        var selectControl = document.getElementById('streets');
        for (loopIndex = 0; loopIndex < options.length; loopIndex++ ){

selectControl.options[loopIndex] = new Option(options[loopIndex].firstChild.data);
        }
    }
    </script></head>
    <body><h1>Map of Hanoi city</h1>
    <table>
        <tr>
            <td><font size="4">Choose District</font></td>
            <td>
                <select onChange="GetDistrict(this.value)" id="district">

<option value="0"></option>

<option value="Hai Ba Trung">Hai Ba Trung</option>

<option value="Hoan Kiem">Hoan Kiem</option>
                </select>
            </td>
        </tr>
        <tr>
            <td><font size="4">Streets</font></td>
            <td>
                <select id="streets">
                </select>
            </td>
        </tr>
    </table></body></html>
    default.php<?php
        header("Content-type: text/xml");
    ?>
    <?xml version="1.0"?>
    <options>
    <?php
    if($_POST["district"]=="Hai Ba Trung"){
        $options = array('Dai Co Viet', 'Tran Khat Chan', 'Bach Mai');
    }
    else if($_POST["district"]=="Hoan Kiem"){
        $options = array('Hang Bai', 'Hang Tre', 'Hang Chieu');
    }

    foreach ($options as $value){
        echo '<option>';
        echo $value;
        echo '</option>';
    }
    ?>
    </options>
```

Bạn có thể thấy rằng, về mặt nguyên tắc, 2 ví dụ được đưa ra trong bài viết đều có cách thức

(Xem tiếp trang 28)

LẬP TRÌNH AJAX...

(Tiếp theo trang 19)

hoạt động tương tự nhau và chỉ khác về mặt xử lí dữ liệu gửi lên cũng như thông tin XML trả về. Vì thế, chỉ xin được giải thích với bạn một số hàm xử lí khác biệt trong 2 ví dụ trên mà thôi.

Trước hết, mời bạn nhìn vào file *default.htm*. Ở đây, ta xét 2 hàm mới được đánh dấu //1 và //2. Tác dụng của dòng lệnh //1 với hàm *open()* là: Gửi thông tin lên file *default.php* trên server thông qua phương thức POST. Còn tác dụng của dòng //2 chỉ đơn thuần báo với server rằng, thông tin đang gửi lên được mã hóa dạng Unicode. Tiếp đó, câu lệnh //3 có sự khác biệt so với trước đây! Thay vì sử dụng hàm *responseText* như ví dụ trên, hàm *responseXML* được dùng với mục đích nhận về thông tin dạng XML! Cuối cùng, ta dùng tiếp hàm *listOptions()* để hiển thị thông tin XML này trên mục Streets được khai báo bên dưới. Như vậy, file *default.htm* đã làm những công việc sau:

- Gửi thông tin mã hóa dạng Unicode lên file *default.php* trên Server theo giao thức POST

- Nhận về thông tin dạng XML bằng hàm *responseXML*, đồng thời, xử lí thông tin XML này thông qua hàm *listOptions()*

Ta lại xét tiếp đến file *default.php*. Nhiệm vụ chính của nó là nhận tên Quận mà Client gửi lên, dựa vào đó, gửi trả lại thông tin dạng XML tương ứng. Vậy, bằng cách nào file PHP có thể chỉ ra cho trình duyệt phía Client hiểu được rằng, thông tin mà nó đang trả về là dạng thức XML? Vâng, thông số *header("Content-type: text/xml");* mà ta khai báo ngay từ đầu có tác dụng giải thích với trình duyệt về điều đó. Bạn bắt buộc phải khai báo thông số này nếu muốn thông tin trả về được định dạng XML, nếu thiếu, trình duyệt chỉ hiểu được rằng đó là dạng tin thuần văn bản! Còn lại, những câu lệnh tiếp theo trong file *default.php* chắc các bạn đều hiểu tác dụng của nó. Rất đơn giản, hàm *echo* chỉ làm nhiệm vụ in ra thông tin trong dấu nháy đơn ('') mà thôi, không có gì phức tạp cả.

Bây giờ, mời bạn quay lại với file *default.htm*. Giả sử ta đã nhận được thông tin XML trả về từ server, hàm *listOptions()* sẽ đảm đương nhiệm vụ hiển thị nó trong danh sách có mã ID là *'streets'*. Tại hàm *listOptions()* này, ta sẽ dùng một vòng lặp để đọc toàn bộ thông tin về các thẻ có tên là *option* được trả về. Bạn có thể hỏi rằng, thẻ *option* này ta lấy về bằng cách nào? Chính câu lệnh //4 làm điều đó! Với hàm *getElementsByTagName()* của đối tượng *XMLHttpRequest*, ta sẽ lấy được một chuỗi nội dung nằm trong thẻ <option> (của thông tin XML được tạo ra từ file *default.php*). Cuối cùng, ta dùng tiếp thuộc tính *firstChild.data* để đọc dữ liệu XML đó ra và gán cho danh sách *'streets'* là hoàn tất ứng dụng!

Bạn thấy đó, trên nguyên tắc, các ngôn ngữ phía Server chỉ làm nhiệm vụ viết ra một file thông tin dạng XML mà thôi. Còn cách truyền thông tin lên server như thế nào, xử lí dữ liệu trả về ra sao tại phía trình duyệt Client thì 2 ứng dụng nhỏ trên chắc cũng phần nào giúp bạn hiểu được đôi chút. Và, xin được hẹn gặp lại các bạn trong những bài viết sau, những bài viết đi sâu hơn về những thủ thuật và "mẹo" nhỏ giúp bạn làm chủ AJAX tốt hơn nữa, thuần thục hơn nữa!

Trần Công Thành