[Bài đọc] Gửi yêu cầu và xử lý kết quả phản hồi AJAX

**Đối tượng XMLHttpRequest**

XMLHttpRequest được sử dụng để truyền tải và thao tác dữ liệu XML tới và từ một WebServer bởi sử dụng HTTP bằng cách thiết lập một kênh kết nối độc lập giữa Client-Side và Server-Side.

**Tạo đối tượng XMLHttpRequest**

nameobj = new XMLHttpRequest();

Với các phiên bản IE cũ hơn (chẳng hạn như IE5 và IE6) sử dụng đối tượng **ActiveXObject**:

nameobj = new ActiveXObject("Microsoft.XMLHTTP");

Trong khi lập trình phát triển các ứng dụng Web, để bao tất cả các trường hợp và để xử lý mọi trình duyệt như trong trường hợp này (để bao cả IE5 và IE6), bạn cần kiểm tra xem trình duyệt có hỗ trợ đối tượng XMLHttpRequest không. Nếu có, bạn tạo một đối tượng XMLHttpRequest; còn nếu không, bạn tạo một đối tượng ActiveXObject:

**var** xhttp;  
**if** (window.XMLHttpRequest){     
 xhttp= **new** XMLHttpRequest();  
}   
**else** {       
 */\*tao doi tuong ActiveXObject neu trinh duyet khong ho tro XMLHttpRequest\*/*     
 xhttp= **new** ActiveXObject("Microsoft.XMLHTTP");  
}

**Gửi yêu cầu AJAX**

Với đối tượng XMLHttpRequest vừa được khởi tạo, bạn có thể sử dụng để gửi yêu cầu tới web server và nhận về các phản hồi. Để gửi yêu cầu tới web server, bạn phải sử dụng kết hợp hai phương thức open() và send() của đối tượng XMLHttpRequest.

Có hai cách cơ bản để gửi yêu cầu AJAX: đồng bộ và không đồng bộ. Khi gửi yêu cầu đồng bộ, đoạn mã gửi yêu cầu sẽ đợi phản hồi từ phía server – quá trình này được gọi là quá trình chặn (blocking). Vì vậy, với yêu cầu đồng bộ, đoạn mã gửi yêu cầu sẽ chặn việc thực thi các mã JavaScript khác trong khi đợi phản hồi từ web server. Quá trình này bộc lộ những nhược điểm rõ ràng khi các yêu cầu hoặc phản hồi bị thất lạc trong quá trình truyền nhận hoặc đơn giản là bị chậm. Với yêu cầu không đồng bộ, dòng mã gửi yêu cầu không chặn các mã khác. Thay vào đó, nó sẽ kiểm tra trạng thái của yêu cầu để biết khi nào yêu cầu hoàn thành. Bạn sẽ tìm hiểu thêm về yêu cầu không đồng bộ trong phần sau.

Trước khi gửi một yêu cầu tới server, bạn cần tạo nó. Phương thức open được sử dụng để tạo yêu cầu, phương thức này có ba đối số: phương thức gửi yêu cầu (GET, POST, HEAD hoặc khác), địa chỉ URL  chứa địa chỉ của trang web mà bạn sẽ gọi tới và một giá trị boolean true hoặc false cho biết bạn muốn gửi yêu cầu theo chế độ không đồng bộ hay đồng bộ.

Giả sử đối tượng XMLHttpRequest được truy xuất thành công thông qua hàm readyAJAX và lưu vào biến requestObj, khi đó lời gọi không đồng bộ tới phương thức open sẽ như sau:

var url = "ajax\_info.txt";

xhttp.open("GET", url, true);

Tương tự lời gọi đồng bộ sẽ như sau:

var url = "ajax\_info.txt";

xhttp.open("GET", url, false);

Quá trình gửi yêu cầu thực sự được thực hiện khi phương thức send được gọi:

xhttp.send();

**Xử lý kết quả phản hồi AJAX**

Sẽ rất dễ dàng nếu làm việc với một yêu cầu đồng bộ vì toàn bộ mã JavaScript sẽ tạm ngừng thực thi cho đến khi có phản hồi. Biến requestObj cung cấp một phương thức hữu ích trong việc xử lý kết quả trả về bằng cách cung cấp luôn mã trạng thái và mô tả trạng thái gửi về từ server. Tuy vậy, bất kể là yêu cầu đồng bộ hay không đồng bộ, mã trạng thái cũng cần được kiểm soát để chắc chắn là toàn bộ quá trình đã diễn ra thành công (thông thường mã trạng thái có giá trị 200 nếu thành công).

**Thuộc tính responseText chứa toàn bộ văn bản trả về từ web server**

Ví dụ, giả sử ứng dụng phía server thực hiện việc trả về toàn bộ nội dung file .txt. Toàn bộ mã của yêu cầu đồng bộ và xử lý kế quả phản hồi trả về từ server như sau:

<!DOCTYPE html>

<html>

<body>

<div id="demo">

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Change Content</button>

</div>

<script>

function loadDoc() {

   var xhttp = new XMLHttpRequest();

   xhttp.open("GET", "ajax\_info.txt", false);

   xhttp.send();

   if (xhttp.status == 200) {

      document.getElementById("demo").innerHTML = xhttp.responseText;

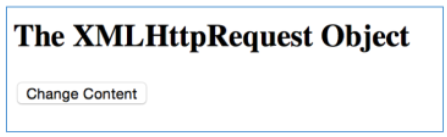
   }

}

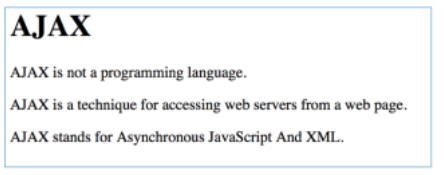
</script>

</body>

</html>



Khi client nhận được phản hồi từ phía web server, nó sử dụng thuộc tính status để kiểm tra trạng thái của dữ liệut trả về. Nếu giá trị nhận được là 200, mọi thứ sẽ suôn sẻ, dữ liệu được lưu trong file ajax\_info.txt sẽ hiển hiện trên màn hình khi button được nhấn.



Việc xử lý các yêu cầu không đồng bộ hơi phức tạp hơn một chút. Khi gửi một yêu cầu loại này, các đoạn mã lệnh phía sau vẫn tiếp tục chạy. Do đó, sẽ rất khó xác định thời điểm mà phản hồi được nhận đầy đủ. Để kiểm soát được chính xác trạng thái của quá trình phản hồi từ server, chúng ta có thể sử dụng sự kiện onreadystatechange để kích hoạt đoạn mã kiểm tra sự thay đổi của thuộc tính readyState của sự kiện này để xác định việc này. Có 5 trạng thài của readystate 0: chưa được khởi tạo, đã mở nhưng chưa được gọi tới. 1: Mở. Đã được khởi tạo nhưng chưa được gửi. 2: Đã gửi. Yêu cầu đã được gửi. 3: Đang nhận. Đang nhận phản hồi. 4: Đã được tải. Đã nhận xong phản hồi.

Trên thực tế, trạng thái duy nhất của thuộc tính readyState mà người lập trình JavaScript và AJAX quan tâm đó là 4 – Đã được tải. Việc thực hiện xử lý các phản hồi có giá trị readyState khác 4 sẽ có thể gây ra lỗi.

Thông thường bạn sẽ sử dụng một hàm đồng bộ để xử lý sự kiện onreadystatechange đối với các lời gọi AJAX đồng bộ. Hàm này thực hiện kiểm tra thuộc readyState có bằng 4 hay không, sau đó kiểm tra để đảm bảo trạng thái phản hồi bằng 200, tức là yêu cầu thành công.

Đoạn mã lệnh trên sửa thành:

<!DOCTYPE html>

<html>

<body>

<div id="demo">

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Change Content</button>

</div>

<script>

function loadDoc() {

  var xhttp = new XMLHttpRequest();

  xhttp.onreadystatechange = function() {

    if (this.readyState == 4 && this.status == 200) {

      document.getElementById("demo").innerHTML = this.responseText;

    }

  };

  xhttp.open("GET", "ajax\_info.txt", true);

  xhttp.send();

}

</script>

</body>

</html>