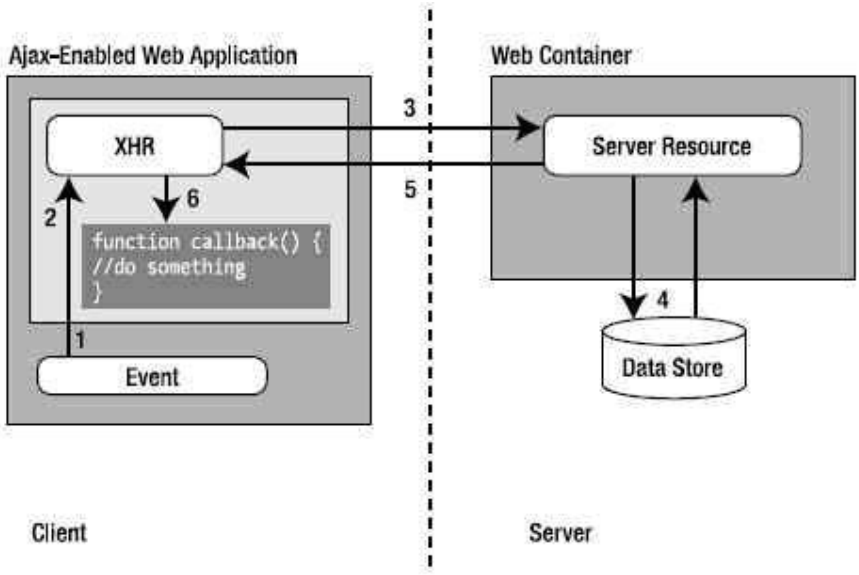
AJAX - Tương lai của ứng dụng Web

Bài 12: Đối tượng XMLHttpRequest - Phân tích các đặc tính - Sự tương tác

Sự tương tác

Ta xét một ví dụ để tìm hiểu các tương tác của Ajax. Hình sau cho thấy mô hình tương tác chuẩn trong một ứng dụng Ajax.



Không giống như các cách tiếp cận kiểu request/response thông thường trong các chuẩn Web client, một ứng dụng Ajax có những khác biệt, sau đây là mô tả quá trình tương tác:

1. Một event client-side gây ra một sự kiện - Ajax event. Bất kỳ một tác động nào cũng có thể gây ra Ajax event, từ một sự kiện onchange đơn giản cho đến một số tác động của người dùng. Ví dụ với đoạn mã sau:

Trích:

```
<input type="text" id="email" name="email" onblur =
"validateEmail()";>
```

2. Một thể hiện của XMLHttpRequest được tạo ra. Dùng phương thức open(), tạo lời gọi hàm - địa chỉ URL được thiết lập cùng với phương thức HTTP yêu cầu, thông thường là GET hay

POST. Request được tạo ra qua việc gọi phương thức send(). Đoạn mã nguồn sau thể hiện điều đó:

Trích:

```
var xmlHttp;
function validateEmail() {
var email = document.getElementById(”email”);
var url = “validate?email=” + escape(email.value);
if (window.ActiveXObject) {
xmlHttp = new ActiveXObject(”Microsoft.XMLHTTP”);
}
else if (window.XMLHttpRequest) {
xmlHttp = new XMLHttpRequest();
}
xmlHttp.open(”GET”, url);
xmlHttp.onreadystatechange = callback;
xmlHttp.send(null);
}
```

3. Một request được tạo và gửi đến server. Có thể là một lời gọi tới một servlet, một CGI script, hay một công nghệ phía server nào đó tương tự như ASP.NET, JSP, hay PHP…

4. Server xử lí các yêu cầu, chẳng hạn như truy cập cơ sở dữ liệu hay một tác vụ hệ thống nào đấy.

5. Response được trả về cho trình duyệt. Trường Content-Type được thiết lập ở dạng text/xml; XMLHttpRequest chỉ có thể xử lí kết quả dạng text/html. Trong các thể hiện phức tạp hơn, response khá rắc rối và bao gồm JavaScript, các thao tác trên đối tượng DOM, hoặc các công nghệ liên quan khác. Chú ý là cũng cần thiết lập header vì thế trình duyệt sẽ không lưu kết quả một cách cục bộ. Ta sẽ làm như sau:

Trích:

```
response.setHeader(”Cache-Control”, “no-cache”);
response.setHeader(”Pragma”, “no-cache”);
```

6. Trong ví dụ sau, cấu hình XMLHttpRequest để gọi hàm callback() khi kết quả xử lí được trả về. Hàm này kiểm tra thuộc tính readyState trên đối tượng XMLHttpRequest và sau đó xem xét mã trạng thái trả về từ server. Mọi thứ hoàn toàn bình thường, hàm callback() có thể làm nhiều việc trên phía client. Một phương thức callback thường có dạng sau:

Trích:

```
function callback() {
if (xmlHttp.readyState == 4) {
if (xmlHttp.status == 200) {
//do something interesting here
}
}
}
```

Có một số khác biệt với mô hình request/response thông thường nhưng không quá lạ lẫm đối với các lập trình viên Web. Rõ ràng, phải xem xét thêm về việc tạo và thiết lập một đối tượng XMLHttpRequest và sau đó (hàm) callback sẽ kiểm tra các trạng thái. Thường thì các lời gọi chuẩn này được đóng gói vào một thư viện để dùng trong ứng dụng, hay nói cách khác là dùng một thư viện có sẵn để thực thi Ajax cho ứng dụng Web (có rất nhiều thư viện như thế, ta sẽ xét trong các phần sau). Ajax là vấn đề tuy còn mới mẻ, nhưng đã có một lượng đáng kể các thư viện và ứng dụng mã nguồn mở được công bố.

Hầu hết các framework và toolkit Ajax trên các trang Web đều dùng các kĩ thuật cơ bản và trừu tượng hóa các trình duyệt, và thêm vào một số component giao diện người dùng (UI). Một số là các framework thuần client; còn lại làm việc trên server. Nhiều framework trong số này mới được bắt đầu xây dựng, nhưng chúng liên tục có các phiên bản và có thên các thư viện mới. Một số giải pháp để thực thi Ajax là các thư viện Ajax.NET, Atlas, libXmlRequest, RSLite, sarissa, JavaScript Object Notation (JSON), JSRS, Direct Web Remoting (DWR), và Ruby on Rails…

Bài sau chúng ta sẽ phân tích "Các phương thức GET và POST".

Bài 13: Đối tượng XMLHttpRequest - Phân tích các đặc tính - GET & POST

Các phương thức GET và POST

Trên lý thuyết, sử dụng GET khi request không thay đổi giá trị, tức là nhiều request sẽ trả về cùng kết quả. Trong thực tế, nếu phương thức tương ứng ở server thay đổi trạng thái theo một vài cách, thì điều này không còn đúng nữa.

Điều này có nghĩa, nó là một chuẩn. Có rất nhiều sự khác biệt với chuẩn trong điều kiện kích thước của phần đệm (payload) - trong nhiều trường hợp, các trình duyệt và server sẽ giới hạn độ dài của địa chỉ URL sử dụng để gửi dữ liệu tới server. Tóm lại, dùng GET để truy lục dữ liệu từ server; hay nói cách khác tránh được việc thay đổi trạng thái trên với lời gọi GET.

Phương thức POST được dùng khi muốn thay đổi trạng thái trên server. Không giống như GET, phải thiết lập phần Content-Type header trên đối tượng XMLHttpRequest như sau:

Trích:

```
xmlHttp.setRequestHeader("Content-Type","application/
x-www-form-urlencoded");
```

POST không hạn chế kích thước của payload được gửi tới server, và POST request không cần bảo đảm tính không đổi.

Hầu hết các request được thiết lập ở GET request; tuy nhiên trạng thái POST cũng luôn sẵn sàng khi cần thiết.

Bài 14: Đối tượng XMLHttpRequest - Remote Scripting - Giới thiệu

Remote Scripting

Về cơ bản, remote scripting là một loại lời gọi các thủ tục từ xa. Sự tương tác với server vẫn như các ứng dụng Web thông thường, nhưng không tái nạp (refresh) toàn bộ trang web. Chỉ với AJAX, mới có thể sử dụng công nghệ bất kỳ phía server để có thể nhận các request, xử lí chúng và trả về kết quả. Với mỗi công nghệ phía server, có một số lựa chọn cho phía client để thực hiện remote scripting.

Có thể nhúng vào các đoạn Flash, Java applet, hay các ActiveX vào ứng dụng. Thậm chí cũng có thể dùng một số công nghệ như XML-RPC, nhưng sự phức tạp của phương pháp này làm giảm tính phổ biến của nó. Cách thực thi cơ bản đối với remote scripting bao gồm phối hợp một cripting với một IFRAME và server trả về các đoạn mã JavaScript, các đoạn mã này sẽ được chạy trong trình duyệt.

Microsoft có giải pháp riêng về remote scripting, được gọi là Microsoft Remote Scripting (MSRS), cho phép gọi các server script giống như là chúng cục bộ. Một Java applet được nhúng vào trang web để làm cho sự liên lạc với server được dễ dàng, một trang asp được dùng để chứa các script phía server, một file .html quản lí bên phía client.

Có thể dùng các giải pháp của Microsoft với Netscape và Internet Explorer 4.0 trở nên. Các lời gọi hàm này có thể đồng bộ hay bất đồng bộ. Tuy nhiên giải pháp này yêu cầu Java, có nghĩa là cần có sự cài đặt thêm vào, và nó phụ thuộc vào trình Internet Information Services (IIS), từ đó làm giảm sự lựa chọn cho phía server.

Ví dụ về Remote Scripting

Để so sánh, chúng ta cùng xét một ví dụ về công nghệ tương tự AJAX được thực thi như thế nào dùng IFRAME. Ví dụ sau chỉ ra việc dùng IFRAME cho remote scripting.

Trong ví dụ này có hai file iframe.html và server.html. Server.html giả lập một response được trả về từ server.

iframe.html

Trích:

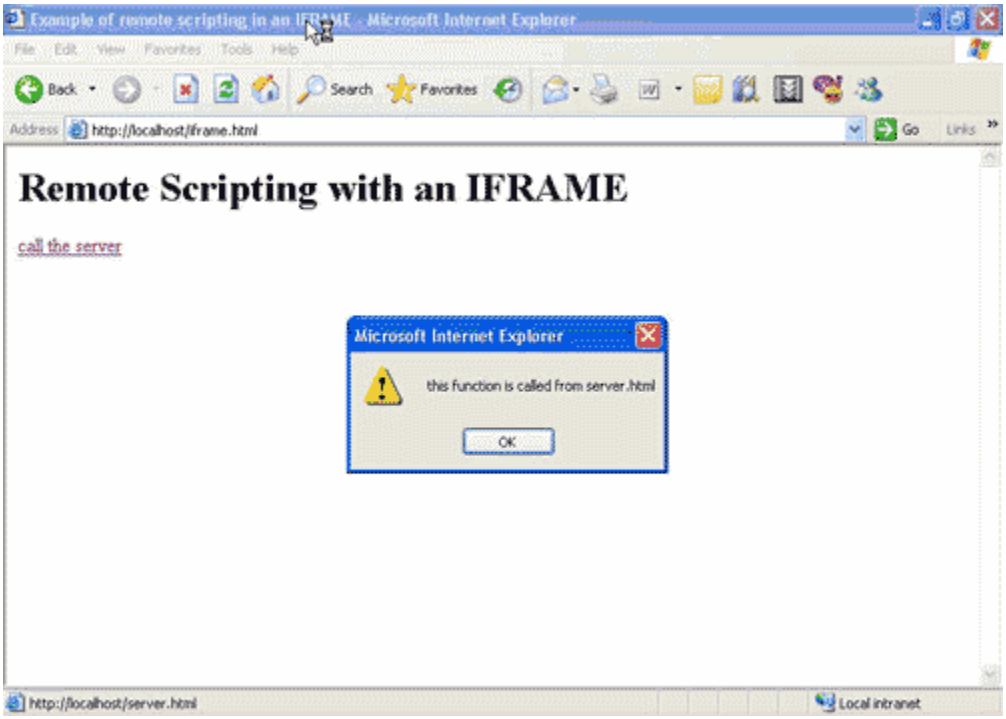
```
<html>
<head>
<title>Example of remote scripting in an IFRAME</title>
</head>
<script type=”text/javascript”>
function handleResponse() {
alert(’this function is called from server.html’);
}
</script>
<body>
<h1>Remote Scripting with an IFRAME</h1>
<iframe id=”beforexhr”
name=”beforexhr”
style=”width:0px; height:0px; border: 0px”
src=”blank.html”></iframe>
<a href=”server.html” target=”beforexhr”>call the server </a>
</body>
</html>
```

server.html

Trích:

```
<html>
<head>
<title>the server</title>
</head>
<script type=”text/javascript”>
window.parent.handleResponse();
</script>
<body>
```

```
</body>
</html>
```


Example of remote scripting in an IFRAME - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/iframe.html
Go Links
Remote Scripting with an IFRAME
call the server
Microsoft Internet Explorer
this function is called from server.html
OK
http://localhost/server.html
Local intranet