Mục lục

[1 Giới thiệu về AJAX 3](#_Toc278931538)

[1.1 AJAX là gì? 3](#_Toc278931539)

[1.2 Nguyên lý hoạt động 3](#_Toc278931540)

[1.3 So sánh với ứng dụng web truyền thống 5](#_Toc278931541)

[2 Sử dụng AJAX 6](#_Toc278931542)

[2.1 XMLHttpRequest 6](#_Toc278931543)

[2.1.1 Định nghĩa 6](#_Toc278931544)

[2.1.2 Tính chất 6](#_Toc278931545)

[2.1.3 Các phương thức của đối tượng XMLHttpRequest 7](#_Toc278931546)

[2.1.4 Các thuộc tính của đối tượng XMLHttpRequest 7](#_Toc278931547)

[2.2 Send request – GET/POST 8](#_Toc278931548)

[2.2.1 Định nghĩa 8](#_Toc278931549)

[2.2.2 Các tham số của phương thức open( ) 8](#_Toc278931550)

[2.3 Server response 11](#_Toc278931551)

[2.4 Kết hợp AJAX với jQuery 12](#_Toc278931552)

[2.4.1 Nhóm hàm dùng đăng ký hàm xử lý sự kiện 12](#_Toc278931553)

[2.4.2 jQuery.ajaxSetup(options) 13](#_Toc278931554)

[2.4.3 Nhóm hàm dùng lấy dữ liệu từ server 13](#_Toc278931555)

[2.4.4 Nhóm hàm dùng tuần tự hóa dữ liệu (để truyền nhận) 17](#_Toc278931556)

[3 AJAX Control Toolkit 19](#_Toc278931557)

[3.1 AJAX Control Toolkit? 19](#_Toc278931558)

[3.2 Hướng dẫn cài đặt 19](#_Toc278931559)

[3.2.1 Môi trường: 19](#_Toc278931560)

[3.2.2 Cài đặt 19](#_Toc278931561)

[3.3 Hướng dẫn sử dụng một số control thông dụng 22](#_Toc278931562)

[3.3.1 ToolkitScriptManager 22](#_Toc278931563)

[3.3.2 Accordion 23](#_Toc278931564)

[3.3.3 Animation 24](#_Toc278931565)

[3.3.4 AlwaysVisibleControlExtender 25](#_Toc278931566)

[3.3.5 AutoComplete 25](#_Toc278931567)

[3.3.6 CalendarExtender 25](#_Toc278931568)

[3.3.7 ColorPickerExtender 26](#_Toc278931569)

[3.3.8 ConfirmButonExtender 26](#_Toc278931570)

[3.3.9 FilteredTextboxExtender 27](#_Toc278931571)

[3.3.10 HoverMenuExtender 28](#_Toc278931572)

[3.3.11 PasswordStrength 29](#_Toc278931573)

[3.3.12 Slideshow 30](#_Toc278931574)

[4 Ưu và khuyết điểm: 31](#_Toc278931575)

[4.1 Ưu điểm: 31](#_Toc278931576)

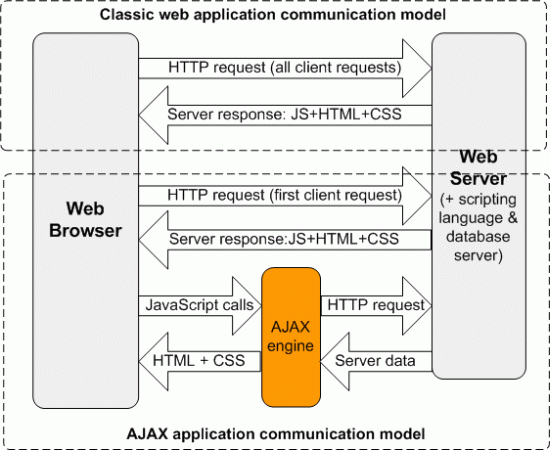
[4.2 Khuyết điểm: 31](#_Toc278931577)

# Giới thiệu về AJAX

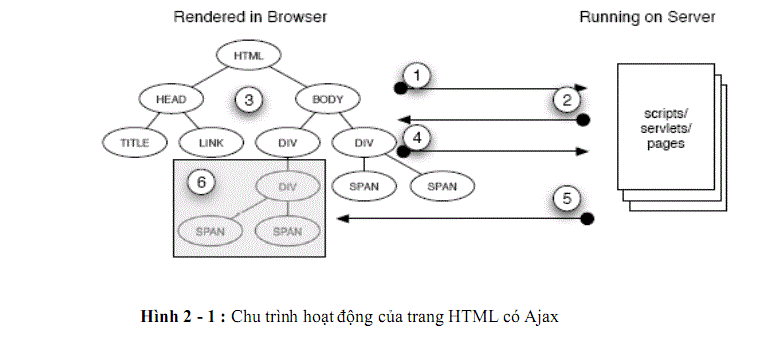
## AJAX là gì?

Ajax là chữ viết tắt của Asynchronous Javascript and XML. Đó không phải là một ngôn ngữ hay một công nghệ riêng lẻ mà là tập hợp của nhiều công nghệ được dùng trong phát triển ứng dụng web (HTML, CSS, XML, DOM, Java script, XMLHttpRequest). Với Ajax dữ liệu có thể lấy từ server một cách bất đồng bộ bên dưới ứng dụng mà không làm ảnh hưởng tới thao tác hiện tại trên trang hiện thời. Sử dụng Ajax đem đến sự linh động trong tương tác, và các giao diện động trên trang web hiệu quả hơn.

## Nguyên lý hoạt động

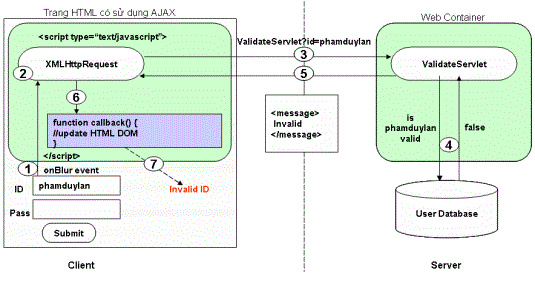


**So sánh nguyên lý hoạt động của ứng dụng web có sử dụng AJAX và ứng dụng web truyền thống**



**Nguyên lý hoạt động của trang web có sử dụng AJAX**

1. Client gửi yêu cầu cho server.
2. Request đầu tiên, server sẻ trả về toàn bộ trang HTML ban đầu (bao gồm JS+HTML+CSS)
3. Trình duyệt sẽ hiển thị trang HTML với cấu trúc cây DOM được lưu trong bộ nhớ.
4. Những thao tác của người dùng tạo nên những yêu cầu “con” gửi tới server theo kiểu bất đồng bộ. Lúc này, cây DOM vẫn không bị thay đổi.
5. Server trả dữ liệu theo những yêu cầu nhận được ở bước 4.
6. Trình duyệt sẽ phân tích kết quả nhận được và cập nhật lại cây DOM với dữ liệu mới, sau đó cập nhật lại giao diện người dùng(trang chỉ được làm mới một phần chứ ko phải toàn bộ trang – “redraw but not refreshed”).

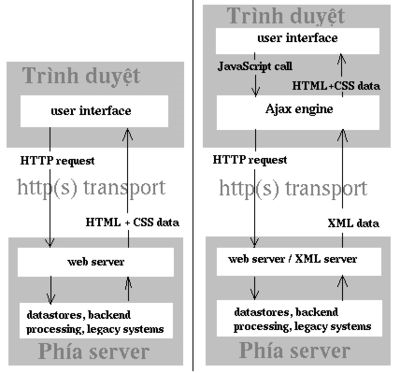
**Ví dụ:**

**Ví dụ minh họa trang HTML có sử dụng AJAX**

1. Khi người sử dụng gõ tên đăng nhập của mình vào trường nhập **ID** xong và chuyển sang t Pass để nhập password, thì ngay lập tức sự kiên **onBlur** được sinh ra, sự kiện này phát sinh khi người dùng chọn 1 phần tử các của form đề nhập dữ liệu làm cho phần tử trước đó bị mất “focus”.
2. Sau đó một đối tượng **XMLHttpRequest** được tạo ra.
3. Đối tượng XMLHttpRequest gửi 1 yêu cầu xác thực tên đăng nhập tới server.
4. Yêu cầu này được xử lý bởi servlet **ValidateServlet**. Servlet này sẽ truy xuất cơ sở dữ liệu để xác thực xem tên đăng nhập “phamduylan” được đăng ký chưa.
5. Servlet ValidateServlet gửi trả kết quả dưới định dạng XML, chỉ ra rằng tên đăng nhập không đúng.
6. Đối tượng HttpRequest gọi hàm callback() để xử lý kết quả nhận được từ server.
7. HTML DOM được cập nhật và hiển thị ra giao diện báo với người dùng tên đăng nhập không đúng.

## So sánh với ứng dụng web truyền thống

Điểm khác biệt với web truyền thống:



Trên tư tưởng điểm khác biệt giữa web có Ajax và web truyền thống là khi có thay đổi cần load lại trang web thì Ajax sẽ tải về những phần cần thay đổi và thay đổi phần đó trên máy người dùng, những phần còn lại vẫn giữ nguyên. Còn web truyền thống thì tải về toàn bộ trang web bao gồm phần cần thay đổi và phần không cẩn thay đổi.

Lợi ích so với web thường:

* Bandwidth : client không load toàn bộ trang web nên việc thực hiện các thao tác nhanh hơn và dữ liệu load về ít hơn.
* Less server process: vì server chỉ làm một vài thao tác ứng với việc update một trang web , nên server xử lý ít hơn tốn it resource hơn.
* Less client process: tương tự brower client chỉ làm một số công việc cần thiết cho việc update nên ít xử lý hơn. Ví dụ đơn giản là không cần load lại các hình ảnh, mà việc render các hình ảnh này cũng tốn resource của client.
* Faster development: việt tạo thư viện riêng biệt xừ lý các sự kiện nhỏ sẽ giúp developer tập trung và làm việc tốt hơn.
* Ease & fast to use: với các lợi ích trên khi thực hiện thao tác thì kết quả sẽ trả về nhanh hơn nên tiện cho việc sử dụng.

# Sử dụng AJAX

## XMLHttpRequest

XMLHttpRequest là thành phần cốt lõi của AJAX. Mọi thác tác trao đổi dữ liệu giữa client và server sử dụng AJAX đều thông qua đối tượng này.

### Định nghĩa

XMLHttpRequest là một đối tượng để truyền nhận và xử lý dữ liệu dạng XML hoặc TEXT bằng giao thức HTTP giữa máy chủ và máy khách.

Tạo mới một đối tượng XMLHttpRequest

if (window.XMLHttpRequest)  
  {// code for IE7+, Firefox, Chrome, Opera, Safari  
  xmlhttp=new XMLHttpRequest();  
  }  
else  
  {// code for IE6, IE5  
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  
  }

### Tính chất

XMLHttpRequest hỗ trợ giao tiếp đồng bộ và bất đồng bộ, cho phép lập trình các phương thức đáp ứng khi đối tượng thay đổi trạng thái.

* Bạn có thể cập nhập lại trang web mà không cần tải lại trang web
* Yêu cầu và nhận dữ liệu từ server khi trang đã load xong.
* Gửi dữ liệu đến 1 máy chủ trong background.
* Dữ liệu trả về phải là text/xml.
* Trang web hiện hành vẫn tiếp tục phát sinh các sự hiện trong khi đối tượng XMLHttpRequest chạy nền bên dưới.
* Chỉ có phép một số lượng giới hạn request được gửi lên.
* Cho phép ta chỉ định một hàm xử lý ứng với sự thay đổi của các trạng thái, hàm xử lý được kích hoạt khi request ở các trạng thái: khởi tạo, bắt đầu, đang trả về, hoàn tất.

1 đối tượng XMLHttpRequest gồm 4 trạng thái khác nhau:

* Inittialized (khởi tạo)
* Started(bắt đầu)
* In the process of being returned( đang trong quá trình xử lí)
* Completely (hoàn thành)

### Các phương thức của đối tượng XMLHttpRequest

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| abort() | Hủy bỏ yêu cầu |
| getAllResponseHeaders() | Hoàn trả về tập hợp các tiêu đề (nhãn hiệu và giá trị) dưới dạng chuỗi |
| getResponseHeader(" *headerLabel* ") | Trả về giá trị chuỗi của một tiêu đề duy nhất |
| open(" *method* ", " *URL* "[, *asyncFlag* [, " *userName*"[, " *password* "]]]) | Gán URL đích, phương thức, và thuộc tính tùy chọn khác của một yêu cầu đang chờ |
| send( *content* ) | Truyền các yêu cầu, tùy chọn với chuỗi postable hoặc dữ liệu đối tượng DOM |
| setRequestHeader(" *label* ", " *value* ") | Gán một nhãn cho tin được gửi đi |

### Các thuộc tính của đối tượng XMLHttpRequest

|  |  |
| --- | --- |
| **Thuộc tính** | **Mô tả** |
| onreadystatechange | Hàm xử lý sự kiện khi đối tượng thay đổi trạng thái. |
| readyState | Lấy trạng thái hiện tại của đối tượng:  0 = đối tượng vừa được tạo mới nhưng chưa được khởi gán giá trị thuộc tính  1 = request đã được mở nhưng phương thức send chưa được gọi thực hiện 2 = phương thức send đã được gọi nhưng chưa chưa có dữ liệu 3 = đã nhận được một vài dữ liệu nhưng phương thức responseText và responseXML chưa thể gọi thực hiện 4 = tất cả dữ liệu đã được truyền xong |
| responseText | Lấy nội dung dữ liệu được trả về từ máy chủ dưới dạng chuỗi |
| responseXML | Lấy nội dung dữ liệu trả về dưới dạng XML DOM |
| status | Lấy mã trạng thái HTTP của đối tượng chẳng hạn như 404 cho "Not Found" hay 200 cho "OK" |
| statusText | Lấy chuỗi trạng thái HTTP của đối tượng |
| timeout | Gets/sets time-out |

## Send request – GET/POST

### Định nghĩa

Để gửi một yêu cầu đến server, ta thường dùng phương thức open() và send() của đối tượng XMLHTTPRequest.

|  |  |
| --- | --- |
| **Phương thức** | **Mô tả** |
| open(method, url, async) | Chỉ ra loại yêu cầu, đường dẫn, và có đồng bộ hay không   * method: loại request: GET hay POST * url: đường dẫn của file trên server * async: true (bất đồng bộ) hay false (đồng bộ) |
| send(string) | Gửi yêu cầu đến server  string: Chỉ được dùng cho POST request |

### Các tham số của phương thức open( )

#### GET hay POST?

GET và POST khác nhau như thế nào và nên dùng vào khi nào? Trên lý thuyết, sử dụng GET khi request không làm thay đổi giá trị, nghĩa là khi có nhiều request đều sẽ trả về kết quả như nhau. Tóm lại, ta dùng GET để truy xuất dữ liệu từ server, hay nói cách khác là không làm thay đổi trạng thái trên server.

*Ví dụ:*

<html>

<head>

<script type="text/javascript">

function loadXMLDoc()

{

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

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

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

xmlhttp.open("GET","demo\_get.asp",true);

xmlhttp.send();

}

</script>

</head>

<body>

<h2>AJAX</h2>

<div id="myDiv"></div>

<button type="button" onclick="loadXMLDoc()">Request data</button>

</body>

</html>

Ta dùng POST mỗi khi muốn thay đổi trạng thái trên server. Đó là khi ta cần cập nhật dữ liệu trên server, hay gửi một số lượng lớn dữ liệu đến server do POST không có giới hạn về kích thước dữ liệu được gửi.

*Ví dụ:*

<html>

<head>

<script type="text/javascript">

function loadXMLDoc()

{

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

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

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

xmlhttp.open("POST","demo\_post.asp",true);

xmlhttp.send();

}

</script>

</head>

<body>

<h2>AJAX</h2>

<div id="myDiv"></div>

<button type="button" onclick="loadXMLDoc()">Request data</button>

</body>

</html>

Để POST dữ liệu như một HTML, ta cần phải thiết lập phần Content-Type header trên đối tượng XMLHttpRequest như sau:

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

*Ví dụ:*

<html>

<head>

<script type="text/javascript">

function loadXMLDoc()

{

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

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

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

xmlhttp.open("POST","demo\_post2.asp",true);

xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");

xmlhttp.send("fname=Henry&lname=Ford");

}

</script>

</head>

<body>

<h2>AJAX</h2>

<div id="myDiv"></div>

<button type="button" onclick="loadXMLDoc()">Request data</button>

</body>

</html>

GET đơn giản hơn và nhanh hơn POST, và có thể được sử dụng trong hầu hết các trường hợp.

#### URL

Tham số url trong phương thức open() là một địa chỉ đến một file trên server.

File này có thể là bất kì loại file nào.

*Ví dụ:*

xmlhttp.open("GET","ajax\_test.asp",true);

#### Asynchronous

Để truy cập dữ liệu theo kiểu bất đồng bộ (asynchronous), thì tham số async trong phương thức open() phải được thiết lập là true.

## Server response

* Để lấy response từ server , chúng ta có 2 loại thuôc tính là responseText, responseXML

|  |  |
| --- | --- |
| Response Text | Response XML |
| Lấy chuổi văn bản lấy về từ máy chủ, trả về giá trị kiểu String | Lấy tài liệu XML Dom, trả về giá trị kiểu XML |
| Được sử dụng cho dữ liệu đơn giản | Được sử dụng cho dữ liệu có cấu trúc phức tạp |
| Ví dụ:  document.getElementById("myDiv").innerHTML=xmlhttp.responseText; | xmlDoc=xmlhttp.responseXML; var txt=""; x=xmlDoc.getElementsByTagName("ARTIST"); for (i=0;i<x.length;i++)   {   txt=txt + x[i].childNodes[0].nodeValue + "<br />";   } document.getElementById("myDiv").innerHTML=txt; |

## Kết hợp AJAX với jQuery

Thư viện jQuery cung cấp rất nhiều hàm và phương thức với đầy đủ các tính năng của AJAX. Các hàm và phương thức này cho phép chúng ta lấy dữ liệu từ server về mà không cần phải tải lại cả trang.

### Nhóm hàm dùng đăng ký hàm xử lý sự kiện

#### ajaxComplete( )

Đăng ký handler sẽ được gọi khi request hoàn thành.

*Cú pháp:*

.ajaxComplete(handler (event, XMLHttpRequest, ajaxOptions));

*Ví dụ: hiển thị một thông điệp khi request hoàn thành*

$("#msg").ajaxComplete(function(event,request, settings){

$(this).append("<li>Request Complete.</li>");

});

#### ajaxError( )

Đăng ký hàm xử lý sự kiện khi request đã hoàn tất nhưng có lỗi xuất hiện. Tương tự như ajaxComplete.

*Cú pháp:*

.ajaxError(handler(event, XMLHttpRequest, ajaxOptions, throwError))

#### ajaxSend( )

Đăng ký một hàm được gọi trước khi một request được gửi đi. Tương tự như ajaxComplete.

*Cú pháp:*

.ajaxSend(handler(event, XMLHttpRequest, ajaxOptions, throwError))

#### ajaxStart( )

Đăng ký một hàm được gọi khi một Ajax request đầu tiên được gửi đi. Tương tự như ajaxComplete.

*Cú pháp:*

.ajaxStart(handler(event, XMLHttpRequest, ajaxOptions, throwError))

#### ajaxStop(handler())

Đăng ký một hàm được gọi khi tất cả các request đã hoàn thành. Tương tự như ajaxComplete.

*Cú pháp:*

.ajaxStop(handler())

#### .ajaxSuccess( handler(event, XMLHttpRequest, ajaxOptions) )

Đăng ký một hàm được gọi mỗi khi có 1 request được thực hiện thành công. Tương tự như ajaxComplete.

*Cú pháp:*

.ajaxSuccess(handler(event, XMLHttpRequest, ajaxOptions, throwError))

### jQuery.ajaxSetup(options)

Thiết đặt giá trị mặc định cho các Ajax request sử dụng về sau (thay vì phải thiết đặt cho từng request trong jQuery.ajax() ).

Tham số nhận vào là những cặp khóa/giá trị.

*Ví dụ:*

$.ajaxSetup({

url: "/xmlhttp/",

type: "POST"

});

$.ajax({ data: myData });

### Nhóm hàm dùng lấy dữ liệu từ server

#### jQuery.ajax(settings)

Thực hiện một HTTP request bất đồng bộ.

Tham số *settings* là một tập các cặp khóa/giá trị dùng để cấu hình request. Tất cả những tham số này đều không bắt buộc. Danh sách tham số và cách sử dụng của nó có thể tham khảo tại <http://api.jquery.com/jQuery.ajax/>.

Phương thức $.ajax() được gọi bên trong tất cả các requests gửi bởi jQuery. Phương thức này thường không được gọi trực tiếp bởi vì có nhiều phương thức ở mức độ cao hơn, dễ sử dụng hơn có thể dùng để thay thế nó.

*Ví dụ 1: Tải và thực thi một tập tin javascript*

$.ajax({

type: "GET",

url: "test.js",

dataType: "script"

});

*Ví dụ 2: Lấy phiên bản mới nhất của một trang HTML*

$.ajax({

url: "test.html",

cache: false,

success: function(html){

$("#display").append(html);

}

});

#### jQuery.get()

Lấy dữ liệu từ server sử dụng một HTTP GET request.

*Cú pháp:*

$.get( url, [ data ], [ callback(data, textStatus, XMLHttpRequest) ], [ dataType ] )

Trong đó:

url: địa chỉ URL chứa dữ liệu cần lấy.

data: dữ liệu được gửi kèm request.

callback(data, textStatus, XMLHttpRequest): một hàm được thực thi khi request hoàn tất.

dataType: kiểu dữ liệu mong muốn nhận được từ server.

*Ví dụ 1: request trang “test.php”*

$.get("test.php");

*Ví dụ 2: request trang test.php và gửi kèm một số dữ liệu*

$.get("test.php", { name: "John", time: "2pm" });

#### jQuery.getJSON()

Lấy dữ liệu được định dạng theo định dạng [JSON](http://www.json.org/) (JavaScript Object Notation) từ server sử dụng một HTTP GET request.

*Cú pháp:*

jQuery.getJSON( url, [ data ], [ callback(data, textStatus, xhr))

Trong đó:

url: địa chỉ URL chứa dữ liệu cần lấy.

data: dữ liệu được gửi kèm request.

callback(data, textStatus, XMLHttpRequest): một hàm được thực thi khi request hoàn tất.

*Ví dụ 1: lấy dữ liệu từ tập tin test.json và hiển thị các thành phần dữ liệu trong đó*

$.getJSON("test.json", function(data) {

$("#display").html("<p>" + data.foo + "</p>" + "<p>" + data.baz[1] + "</p>");

});

//Nội dung tập tin JSON

{

"foo": "The quick brown fox jumps over the lazy dog.",

"bar": "ABCDEFG",

"baz": [52, 97]

}

*Kết quả hiển thị:*

The quick brown fox jumps over the lazy dog.

97

#### jQuery.getScript( )

Tải về một tập tin JavaScript từ server và thực thi nó, sử dụng một GET Http request.

*Cú pháp:*

jQuery.getScript( url, [success(data, textStatus)])

Trong đó:

url: địa chỉ URL chứa dữ liệu cần lấy.

success(data, textStatus): hàm được thực thi khi request thành công.

*Ví dụ: đoạn mã jQuery trong tập tin html sau đây sẽ lấy một tập tin javascript từ một máy chủ trên mạng và và thực thi nó để hiển thị hiệu ứng màu sắc.*

<!DOCTYPE html>

<html>

<head>

<style>.block {

background-color: blue;

width: 150px;

height: 70px;

margin: 10px;

}

</style>

<script src="JQuery.js"></script>

</head>

<body>

<button id="go">&raquo; Run</button>

<div class="block"></div>

<script>$.getScript("http://dev.jquery.com/view/trunk/plugins/color/jquery.color.js", function(){

$("#go").click(function(){

$(".block").animate( { backgroundColor: "pink" }, 1000)

.animate( { backgroundColor: "blue" }, 1000);

});

});</script>

</body>

</html>

#### load( )

Tải dữ liệu từ server và đặt dữ liệu HTML trả về vào element gọi thực hiện phương thức.

*Cú pháp:*

.load( url, [ data ], [ complete(responseText, textStatus, XMLHttpRequest) ] )

Trong đó:

url: địa chỉ URL chứa dữ liệu cần lấy.

data: dữ liệu gửi về server

complete(responseText, textStatus, XMLHttpRequest) ]: hàm được thực thi khi request hoàn thành.

*Ví dụ: lấy dữ liệu từ 1 tập tin html và hiển thị*

$("#btnLoad").click(function () {

$("#displayForLoad").load("TextToGet.html");

});

#### jQuery.post( )

Lấy dữ liệu từ server sử dụng một HTTP POST request.

Tương tự như jQuery.get( ).

### Nhóm hàm dùng tuần tự hóa dữ liệu (để truyền nhận)

#### jQuery.param( )

Chuyển một mảng hay một đối tượng về dạng chuỗi tuần tự, dùng cho việc truyền nhận dữ liệu.

*Cú pháp:*

jQuery.param( obj, traditional )

Trong đó:

Obj: một mảng hay một đối tượng cần chuyển đổi (serialize)

Traditional: biến Boolean cho biết có thực hiện chuyển đổi theo kiểu “shallow” truyền thống hay không

*Ví dụ:*

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>Param demo</title>

<style>div { color:red; }</style>

<script type="text/javascript" src="../JQuery.js" > </script>

</head>

<body>

<div id="results"></div>

<script type="text/javascript">

var params = { width:1680, height:1050 };

var str = jQuery.param(params, true);

$("#results").text(str);

</script>

</body>

</html>

*Kết quả hiển thị:*

width=1680&height=1050

#### serialize( )

Chuyển các element về dạng chuỗi để truyền về server.

*Cú pháp:*

.serialize()

*Ví dụ:*

#### serializeArray( )

Chuyển các element thành một mảng các cặp tên/giá trị.

*Cú pháp:*

.serializeArray()

# AJAX Control Toolkit

## AJAX Control Toolkit?

Là một tập hợp các control được xây dựng sẵn để hỗ trỡ cho việc lập trình ASP.NET và Ajax được phát triển bởi Microsoft.

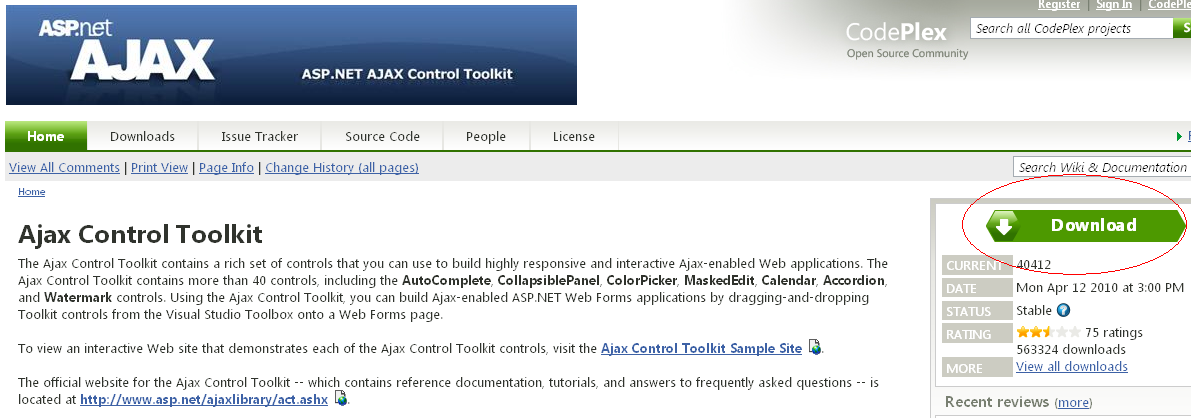
## Hướng dẫn cài đặt

### Môi trường:

* Microsoft Visual Studio 2008
* .NET Framework 3.5

### Cài đặt

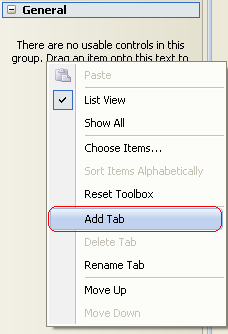
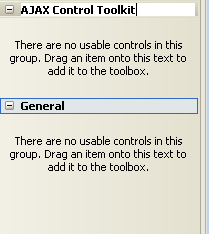
Download AJAX Control Toolkit tại <http://ajaxcontroltoolkit.codeplex.com/>



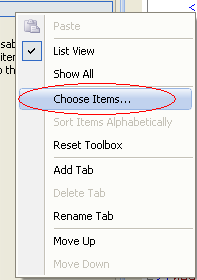
Giải nén thư mục vừa download.

Add thư viện vào VS:

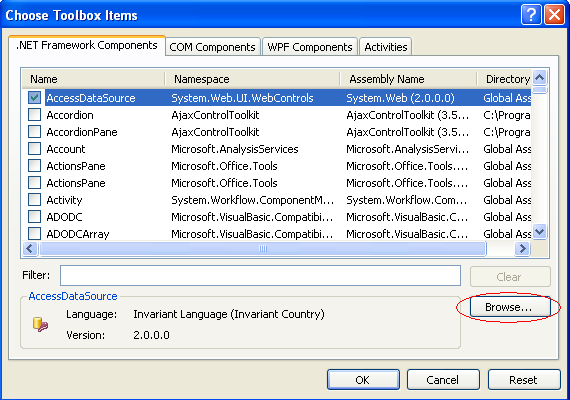
Click phải vào hộp thoại Toolbox của Visual Studio, chọn Add Tab, điền tên tab mới cho toolbox: AJAX Control Toolkit

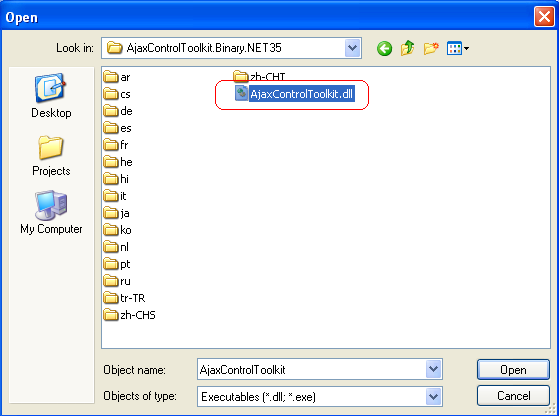
 

Click phải vào tab mới tạo, chọn Choose Items…

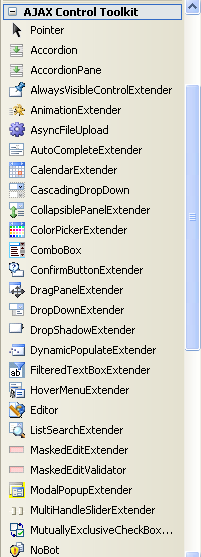


Chọn đường dẫn tới dll





Các control sau khi được thêm vào:

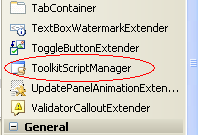


## Hướng dẫn sử dụng một số control thông dụng

### ToolkitScriptManager

Là công cụ quản lý, phân phối và điều khiển tất cả các script được sinh ra bởi việc sử dụng Ajax Control Toolkit.

Muốn sử dụng được tất cả các control của Ajax Control Toolkit đều phải sử dụng ToolkitScriptManager trước.



### Accordion

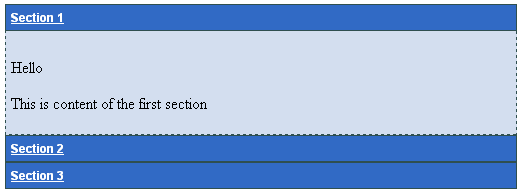
Accordion là một control cho phép cung cấp nhiều vùng hay ô và hiển thị chúng ở cùng một thời điểm. Control này giống dạng như một Panel có thể co giãn nhưng chỉ có một vùng có thể được mở rộng trong một thời điểm.

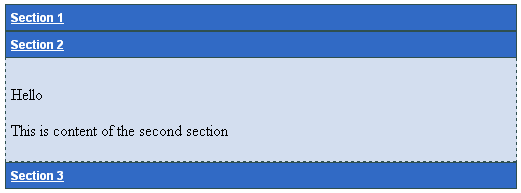
Control Accordion chứa nhiều control AccordionPane, mỗi AccordionPane có tiêu đề (Header) và nội dung (Content) của nó.

Các thuộc tính của Accordion:

* HeaderCssClass: tên của lớp Css được dùng cho tiêu đề.
* HeaderSelectedCssClass: tên của lớp Css được dùng cho tiêu đề sau khi được chọn.
* ContentCssClass: tên của lớp Css được dùng cho phần nội dung.
* FadeTransitions: nếu giá trị true thì dùng để làm mờ dần trong các hiệu ứng chuyển tiếp, nếu là false chuyển tiếp bình thường.
* TransitionDuration: số mili giây chuyển tiếp.
* FramesPerSecond: số khung hình trên giây được sử dụng trong chuyển tiếp hình ảnh.
* Panes: tập hợp chứa các control AccordionPane.
* ….

|  |
| --- |
| <form id="form1" runat="server">  <div>  <asp:ToolkitScriptManager ID="ToolkitScriptManager2" runat="server">  </asp:ToolkitScriptManager>  <asp:Accordion ID="Accordion1" runat="server" HeaderCssClass="accordionHeader"  ContentCssClass="accordionContent">  <Panes>  <asp:AccordionPane ID="AccordionPane1" runat="server" >  <Header>  <a href = "" class = "accordionLink"> Section 1</a>  </Header>  <Content>  <p>Hello</p>  <p>This is content of the first section</p>  </Content>  </asp:AccordionPane>  <asp:AccordionPane ID="AccordionPane2" runat="server">  <Header>  <a href = "" class = "accordionLink"> Section 2</a>  </Header>  <Content>  <p>Hello</p>  <p>This is content of the second section</p>  </Content>  </asp:AccordionPane>  <asp:AccordionPane ID="AccordionPane3" runat="server">  <Header>  <a href = "" class = "accordionLink"> Section 3</a>  </Header>  <Content>  <p>Hello</p>  <p>This is content of the third section</p>  </Content>  </asp:AccordionPane>  </Panes>  </asp:Accordion>  </div>  </form> |





### Animation

Lớp AnimationExtender cho phép sử dụng một framework hoạt hình mạnh mẽ với trang hiện tại một cách dễ dàng. Nó phát hình ảnh động bất cứ khi nào có một sự kiện đặc biệt xảy ra như Onload, OnClick, OnMouseOver, hoặc OnMouseOut được nâng lên bởi control đích.

Framework này cung cấp những hình ảnh động để xử lý chuyển động, thay đổi kích thước, mờ dần, …

|  |
| --- |
| <asp:AnimationExtender ID="OpenAnimation" TargetControlID="btnInfo" runat="server">  <Animations>  <OnClick>  <Sequence>  //Xử lý sự kiện onclick  </Sequence>  </OnClick>  </Animations>  </asp:AnimationExtender> |

Thuộc tính TargetControlID chứa id của control phát sinh animation khi xảy ra sự kiện.



### AlwaysVisibleControlExtender

Bất kể control nào được extend bởi control AlwaysVisibleControlExtender sẽ luôn luôn được hiển thị.

|  |
| --- |
| <asp:ScriptManager ID="ScriptManager1" runat="server">  </asp:ScriptManager>  <ajaxToolkit:AlwaysVisibleControlExtender runat="server"  ID="AlwaysVisibleControlExtender1"  TargetControlID="allwaysOnMessage"  </ajaxToolkit:AlwaysVisibleControlExtender> |

Thuộc tính TargetControlID là id của control sẽ luôn luôn được hiển thị.

### AutoComplete

### CalendarExtender

Control CalendarExtender cho phép hiển thị một hộp thoại chọn ngày. Calendar là một ASP AJAX mở rộng, có thể được gắn vào bất kỳ control Textbox.

Có thể tương tác với control bằng cách bấm vào một ngày để thiết lập ngày mặc định.

Các mũi tên trái, phải được sử dụng để di chuyển qua tháng trước đó hoặc tháng tiếp theo.

|  |
| --- |
| <div>  <asp:ToolkitScriptManager ID="ToolkitScriptManager1" runat="server">  </asp:ToolkitScriptManager>  <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>  <img src=Calendar\_scheduleHS.png runat="server" id="Image"/>  <asp:CalendarExtender ID="CalendarExtender1" runat="server" TargetControlID="TextBox1" PopupButtonID="Image">  </asp:CalendarExtender>  </div> |

### ColorPickerExtender

Control ColorPickerExtender cho phép hiển thị hộp thoại chọn màu sắc. Có thể đính kèm control này với bất cừ control textbox nào của ASP.NET.



Các thuộc tính

* TargetControlID: id của control texbox để mở rộng.
* PopupButtonID: id của control sử dụng để hiển thị bảng chọn màu. Nếu giá trị này không được thiết lập, các bảng chọn màu sẽ xuất hiện khi control Textbox được đại diện bởi TargetControlID nhận được focus.
* SampleControlID: id của control sử dụng để hiển thị màu được chọn, màu nền của control này hiển thị màu mà chuột chọn.
* PopupPosition: chỉ ra nơi popup chọn màu sẽ xuất hiện liên quan tới control textbox đang được mở rộng. Các giá trị có thể: BottomLeft, BottonRight, TopLeft, TopRight, Left hoặc Right. Mặc định là BottomLeft.
* SelectedColor: Giá trị màu được chọn.
* OnClientColorSelectionChanged: một hàm Javascript sẽ được gọi khi các sự kiện colorSelectionChanged được nâng lên.

### ConfirmButonExtender

ConfirmButton là một phần mở rộng bắt sự kiện click vào button, sẽ hiển thị thông báo cho người dùng.

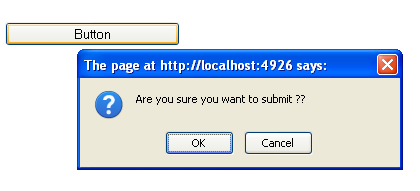
* Nếu người dùng click OK, các chức năng button và liên kết bình thường.
* Nếu người dùng không nhấp OK, sự kiện click bị mắc kẹt và các nút không thực hiện hành vi mặc định của chương trình.
* Người dùng có thể tùy chọn xác định kịch bản client để thực hiện khi các nút trong hộp thoại được xác nhận.

ConfirmButton thường được sử dụng để yêu cầu người dùng xác nhận một hành động nào đó.

Các thuộc tính:

* TargetControlID: Các ID của button để mở rộng.
* ConfirmText: Đoạn văn bản xác nhận để hiển thị. Các thực thể HTML có thể được sử dụng như cho 1 ký tự xuống dòng.
* OnClientCancel: script trên máy client khi người dùng click vào nút Cancel trong hộp thoại xác nhận.
* ConfirmOnFormSubmit: true nếu hộp thoại xác nhận không được hiển thị ngay trước khi form được submit. Điều này hữu ích khi trang ASP.Net chứa các control validator và hộp thoại xác nhận sẽ chỉ được hiển thị sau khi các kiểm tra xác nhận đã thành công.
* DisplayModalPopupID: id của control ModalPopup sử dụng để hiển thị hộp thoại xác nhận thay vì mặc định của window: confirmwindow. Khi sử dụng thuộc tính DisplayModalPopupID, các điều kiện sau phải được đáp ứng:
  + ModalPopup phải được cấu hình với với giá trị TargetControlID như ConfirmButtonExtender..
  + Điều khiển ModalPopup phải ghi rõ OkControlID hoặc CancelControlID để xác nhận các nút tương ứng với các nút OK, Cancel trong hộp thoại xác nhận.
  + ModalPopup không phải chỉ định một thuộc tính OnOkScript hoặc OnCancelScript.

|  |
| --- |
| <asp:ScriptManager ID="ScriptManager1" runat="server" />  <div>  <br />  <asp:Label ID="Label1" runat="server" Width="273px"></asp:Label><br />  <br />  <asp:Button ID="Button1" runat="server" OnClick="Button1\_Click" Text="Button" Width="175px" />  <ajaxToolkit:ConfirmButtonExtender ID="ConfirmButtonExtender1" runat="server" TargetControlID="Button1" ConfirmText="Are you sure you want to submit ??">  </ajaxToolkit:ConfirmButtonExtender>  </div> |



### FilteredTextboxExtender

FilteredTextboxExtender là một control cho phép người dùng chỉ được nhập ký tự mà bạn xác định vào textbox hoặc ngăn cản người dùng nhập ký tự mà bạn chỉ định.

Vì control thực hiện dựa trên Javascript, không nên cho rằng các dữ liệu được gửi tới server chỉ gồm những ký tự bạn cho phép, luôn thực hiện kiểm tra xác nhận ở phía server trên dữ liệu được gửi từ client.

Các thuộc tính:

* TargetControlID: các ID của control cần kiểm tra.
* FilterType: Các loại bộ lọc được áp dụng, quy trình như một sự kết hợp cách nhau bằng dấu phẩy các giá trị: Numbers, LowercaseLetters, UppercaseLetters, và Custom.
* FilterMode: các chế độ lọc được áp dụng, những giá trị được hỗ trợ là ValidChars và InValidChars. Nếu thuộc tính được thiết lập InvalidChars, FilterType phải được thiết lập dạng Custom.
* ValidChars: một chuỗi bao gồm tất cả các ký tự được xem là hợp lệ cho textbox khi kiểu trường là Custom. Nếu kiểu trường không ở dạng Custom, giá trị thuộc tính được bỏ qua.
* InvalidChars: một chuỗi bao gồm tất cả các ký tự được xem là không hợp lệ cho textbox khi kiểu trường là Custom. Nếu kiểu trường không ở dạng Custom, giá trị thuộc tính được bỏ qua.
* FilterInterval: một số nguyên chỉ định khoảng thời gian tính bằng mili giây mà nội dung của các trường được lọc, mặc định là 250.

|  |
| --- |
| <cc1:FilteredTextBoxExtender ID="FilteredTextBoxExtender1"  runat="server" TargetControlID="TextBox1" FilterType="Numbers">  </cc1:FilteredTextBoxExtender>  <cc1:FilteredTextBoxExtender ID="FilteredTextBoxExtender2"  runat="server" TargetControlID="TextBox2" FilterType="LowercaseLetters">  </cc1:FilteredTextBoxExtender>  <cc1:FilteredTextBoxExtender ID="FilteredTextBoxExtender3"  runat="server" TargetControlID="TextBox3" FilterType="Custom, Numbers" ValidChars="+-=/\*().">  </cc1:FilteredTextBoxExtender> |

### HoverMenuExtender

HoverMenuExtender là một control mở rộng mà có thể gắn vào bất kỳ một WebControl Asp.Net và kết hợp control đó với một bảng nội dung hiển thị thêm.

* Các bảng popup được hiển thị ở một vị trí xác định bởi các lập trình viên (trái, phải, trên hoặc phía dưới của control chính).

Các thuộc tính:

* TargetControlID: id của control mà khi xảy ra một sự kiện ở control này thì các popup menu được hiển thị.
* PopupControlID: id của control hiển thị khi chuột được rê lên trên control target.
* HoverCssClass: lớp CSS để áp dụng cho các popup menu khi rê chuột có thể nhìn thấy.
* PopupPosition: Nơi mà popup được bố trí so với TargetControl: trái, phải, trên, dưới, trung tâm.
* OffsetX/OffsetY: số lượng điểm ảnh cho khoảng trống của Popup so với vị trí mặc định của nó.
* HoverDelay: thời gian tính bằng mili giây, trước khi popup hiển thị sau khi lơ lửng trên control target. Mặc định là 0.
* PopDelay: Thời gian tính bằng mili giây, cho popup giữ nguyên sau khi con chuột di chuyển ra khỏi control target. Mặc định là 100.
* Animations: Những animation chung cho HoverMenuExtender.
* OnShow: Những animation hiển thị mỗi khi hovermenu được hiển thị. Các hình ảnh động có thể sử dụng để hiển thị hover menu cùng với các hình ảnh khác.
* OnHide: Các animation OnHide sẽ được phát mỗi lần di chuyển là ẩn menu.

|  |
| --- |
| <cc1:HoverMenuExtender ID="HoverMenuExtender1"  runat="server"  TargetControlID="btnProduct"  PopupControlID="PanelPopUp"  PopupPosition="bottom"  OffsetX="6"  PopDelay="25" HoverCssClass="popupHover">  </cc1:HoverMenuExtender>  <asp:Panel ID="PanelPopUp" runat="server" Height="50px" Width="125px" CssClass="popupMenu">  <asp:LinkButton ID="BtnUpDate" runat="server" CausesValidation="true" CommandName="Update" Text="Update Record"></asp:LinkButton>  <asp:LinkButton ID="BtnCancel" runat="server" CausesValidation="false" CommandName="Cancel" Text="Cancel"></asp:LinkButton>  </asp:Panel>  <asp:LinkButton ID="btnProduct" runat="server">Click here</asp:LinkButton> |

### PasswordStrength

PasswordStrength là một control mở rộng của ASP.NET AJAX được gắn vào một control Textbox được sử dụng cho các mục nhập của mật khẩu. Control PasswordStrength cho biết sức mạnh của mật khẩu trong các textbox và tự cập nhật khi người dùng gõ mật khẩu.

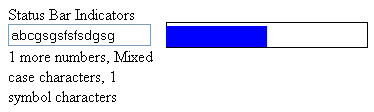
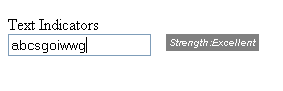
Các chỉ số có thể hiển thị độ mạnh của mật khẩu dưới dạng tin nhắn văn bản hoặc với một thanh hiển thị tiến trình.

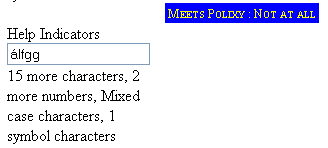
Kiểu và vị trí của cả 2 loại chỉ số có thể cấu hình.

Độ mạnh của mật khẩu cũng có thể cấu hình, cho phép lập trình viên cấu hình độ mạnh của mật khẩu.

Một số thuộc tính:

* TargetControlID: id của textbox để đính kèm vào.
* DisplayPosition: vị trí hiển thị chỉ số độ mạnh của mật khẩu.
* StrengthIndicatorType: loại chỉ số độ mạnh (văn bản hay thanh chỉ thị).
* PreferredPasswordLength: Độ dài mật khẩu được ưu tiên.
* PrefixText: đoạn văn bản tiền tố để hiển thị khi StrengthIndicatorType=text.
* TextCssClass: lớp Css áp dụng khi đoạn văn bản hiển thị khi StrengthIndicatorType=text.
* MinimumNumericCharacters: tối thiểu số ký tự số.
* MinimumSymbolCharacters: tối thiểu ký tự biểu tượng ($ ^ \* …).
* RequiresUpperAndLowerCaseCharacters: nếu trường hợp pha trộn các ký tự được yêu cầu.
* MinimumLowerCaseCharacters: chỉ có hiệu lực nếu có thuộc tính RequiresUpperAndLowerCaseCharacters. Chỉ định số lượng tối thiểu của các ký tự là ký tự thường yêu cầu khi cần trường hợp hỗn hợp như một phần độ mạnh của mật khẩu.
* MinimumUpperCaseCharacters: chỉ có hiệu lực nếu có thuộc tính RequiresUpperAndLowerCaseCharacters. Chỉ định số lượng của các ký tự là chữ hoa cần thiết khicaafn trường hợp hỗn hợp như như một phần độ mạnh của mật khẩu.
* TextStrengthDescriptions: danh sách những mô tả sử dụng khi StrengthIndicatorType=text (nhỏ nhất:2 lớn nhất: 10, trật tự từ yếu tới mạnh nhất).
* CalculationWeightings: danh sách những giá trị có dấu chấm phẩy ngăn cách số được sử dụng để xác định trọng số của một điểm mạnh.
* …





### Slideshow

# Ưu và khuyết điểm:

## Ưu điểm:

* Giao diện trang web được sử dụng hiệu quả hơn, bởi vì chỉ có một phần của trang cập nhật, làm cho người sử dụng cảm thấy kết quả xử lý được hiển thị tức thì, không cần nạp lại trang, giảm đó giảm lượng lớn băng thông và thời gian (load) tải trang về.
* Thời gian chờ đợi giảm - khi khách truy cập gửi một yêu cầu, họ không còn phải chờ đợi cho toàn bộ trang được xây dựng và truyền lại bởi server. Thay vào đó, chỉ có thay đổi nội dung liên quan, và trong trường hợp không quan trọng, khách truy cập vẫn có thể làm việc trong khi dữ liệu đang được gửi.
* Nếu một phần của trang gặp một lỗi, các phần khác không bị ảnh hưởng (nếu chúng không có liên kết logic với nhau) và các dữ liệu đã được nhập bởi người dùng không bị mất.

## Khuyết điểm:

* Xây dựng một ứng dụng AJAX có thể mất nhiều thời gian phát triển và chi phí so với xây dựng một ứng dụng web bình thường, bởi vì đây là công nghệ hỗn hợp nhiều loại.
* Các trang web được tạo động không được ghi vào bộ lưu lịch sử lướt web của trình duyệt, do đó nút "back" (quay lui) của trình duyệt sẽ mất tác dụng quay lại trang thái trước đó của trang sử dụng AJAX, thay vào đó sẽ quay lại trang web trước đó mà người dùng ghé thăm.
* Việc cập nhật các trang web động cũng gây khó khăn cho người dùng trong việc [bookmark](http://vi.wikipedia.org/w/index.php?title=Internet_bookmark&action=edit&redlink=1) (đánh dấu địa chỉ yêu thích) một trạng thái nào đó của ứng dụng.
* Không thể hiện thị nội dung AJAX trên các trang tìm kiếm vì các trang tìm kiếm hiện nay vẫn chưa hỗ trợ tìm vì rất khó tìm và gần như không thể tìm được.
* Mối quan tâm lớn nhất với AJAX là khả năng tiếp cận người dùng. Vì không phải tất cả các trình duyệt có hỗ trợ đầy đủ cho JavaScript hay đối tượng XMLHttpRequest, hoặc đã bị vô hiệu hóa JavaScript. Ngoài ra, khi thiết kế các ứng dụng, vì cách tiếp cận và sử dụng đối tượng XMLHttpRequest trong Internet Explorer và các trình duyệt khác là khác nhau, nên phải xử lý từng trường hợp riêng biệt.
* Việc thiếu các chuẩn cơ bản của Ajax đồng nghĩa với việc không có nhiều sự chọn lựa thực tiễn tốt nhất để kiểm tra các ứng dụng Ajax. Các công cụ kiểm thử cho Ajax thường không hiểu các mô hình sự kiện, mô hình dữ liệu và giao thức của Ajax.
* Bất lợi cuối cùng nằm trong bản thân đối tượng XMLHttpRequest. Do hạn chế bảo mật, bạn chỉ có thể sử dụng nó để truy cập thông tin từ máy chủ mà phục vụ cho trang ban đầu. Nếu bạn cần hiển thị thông tin từ các máy chủ khác, thì AJAX không thể làm được