Chương VII: AJAX

Khoa Công Nghệ Thông Tin & Truyền Thông
Đại học Cần Thơ
Giảng viên: TS. Hà Duy An

Nội dung

I. AJAX

1. AJAX là gì?
2. Cách thức hoạt động
3. XMLHttpRequest Object
4. AJAX & XML File

II. AJAX, PHP, & MySQL

Nội dung

1. AJAX là gì?
2. Cách thức hoạt động
3. XMLHttpRequest Object
4. AJAX & XML File

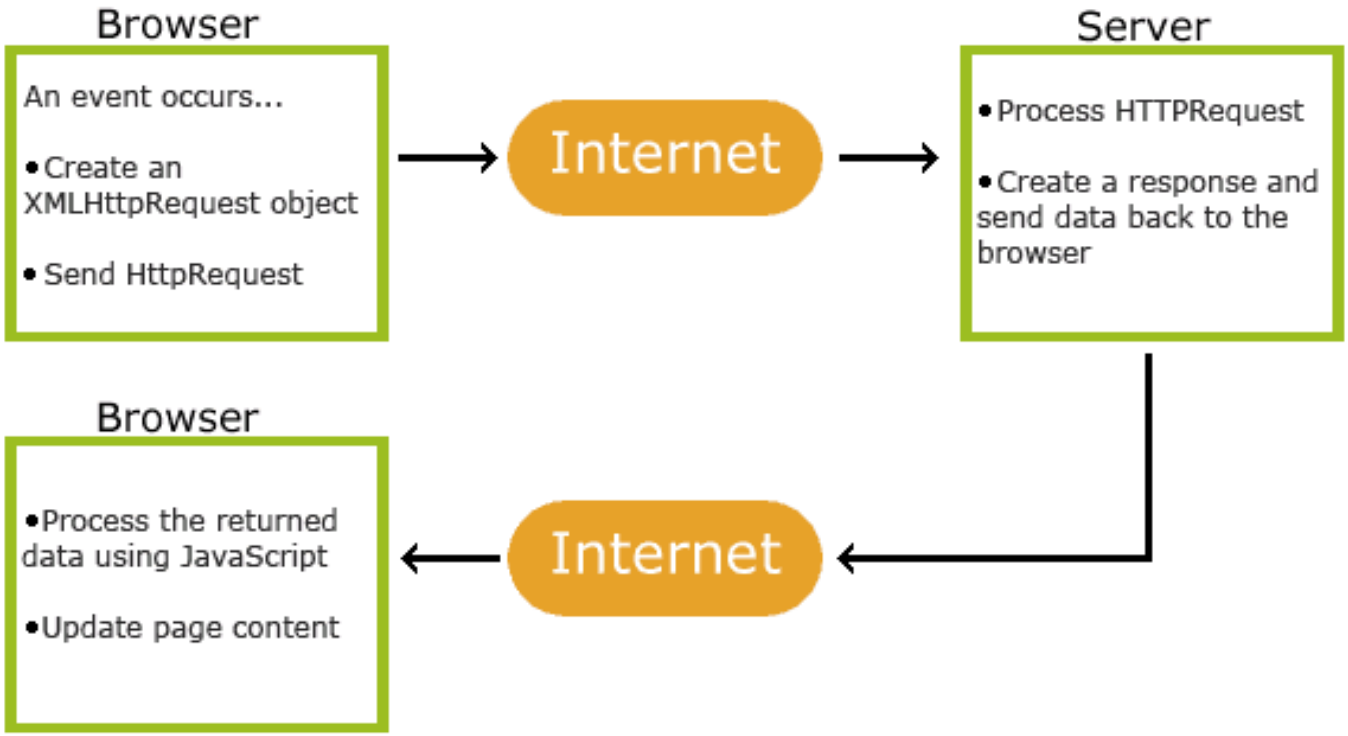
1. AJAX LÀ GÌ?

- AJAX = Asynchronous JavaScript and XML
- AJAX không phải là một ngôn ngữ lập trình mới, nó cung cấp một hướng tiếp cận mới cho các ứng dụng web dựa trên các chuẩn có sẵn
- Đối với các ứng dụng web truyền thống nếu muốn cập nhật dù bất cứ nội dung thông tin gì trên trang web thì bắt buộc trình duyệt phải "refresh" lại trang => gây lãng phí và tốn thời gian chờ đợi không cần thiết của người dùng
- AJAX cho phép tạo ra các trang web có khả năng cập nhật từng phần nội dung của một trang web mà không cần reload lại trang
- Có tính độc lập về nền tảng và ứng dụng
- Được hỗ trợ bởi hầu hết các trình duyệt phổ biến

2. CÁCH THỨC HOẠT ĐỘNG

- AJAX dựa trên các chuẩn Internet, nó là sự kết hợp của:
  - XMLHttpRequest object: trao đổi dữ liệu với server
  - JavaScript + HTML DOM: dùng để hiển thị dữ liệu, tương tác với người dùng
  - CSS: trình bày dữ liệu
  - XML/JSON/Plain text: định dạng dữ liệu cho mục đích truyền dữ liệu
- Với kỹ thuật AJAX dữ liệu được truyền giữa browser và server có thể được ẩn bên dưới browser, có 2 chế độ hoạt động:
  - **Đồng bộ**: trình duyệt sẽ chờ cho đến khi server trả về dữ liệu trước khi các lệnh JavaScript tiếp theo được thực thi
  - **Bất đồng bộ**: các lệnh JavsScript tiếp theo sẽ được thực thi sau khi yêu cầu được gửi mà không chờ kết quả trả về từ server

Mô hình hoạt động



- Nguồn: http://w3schools.com

Truy tài nguyên khác tên miền

- Vì lý do bảo mật, các trình duyệt hiện đại không cho phép truy cập tài nguyên khác tên miền.

- Điều này có nghĩa là cả trang web và tệp XML mà nó cố tải phải được đặt trên cùng một máy chủ.

3. XMLHTTPREQUEST OBJECT

- Trọng tâm của kỹ thuật AJAX là việc sử dụng đối tượng XMLHttpRequest để gửi các yêu cầu và nhận kết quả trả về từ server

- Các bước khởi tạo và sử dụng đối tượng XMLHttpRequest:
  1. Tạo một đối tượng XMLHttpRequest
  2. Định nghĩa một callback function
  3. Mở đối tượng XMLHttpRequest
  4. Gửi yêu cầu đến máy chủ

Sử dụng đối tượng XMLHttpRequest

1. Tạo đối tượng XMLHttpRequest

- Hầu hết các trình duyệt đều hỗ trợ tạo đối tượng XMLHttpRequest
- Cú pháp tạo một đối tượng XMLHttpRequest:

```
xhttp = new XMLHttpRequest();
```

2. Callback Function

- Là một hàm được truyền dưới dạng tham số cho hàm khác.
- Trong trường hợp này, callback function chứa mã để thực thi khi phản hồi sẵn sàng, cú pháp:

```
xhttp.onload = function() {
  // What to do when the response is ready
}
```

- Chỉ cần khi hoạt động ở chế độ bất đồng bộ

Sử dụng đối tượng XMLHttpRequest (tt)

3. Mở đối tượng XMLHttpRequest

- open(*method,url,async*): dùng để xác định phương thức gửi dữ liệu, url và việc xử lý dữ liệu đồng bộ hay bất đồng bộ
  - *method*: phương thức gửi dữ liệu: GET hay POST
  - *url*: vị trí của tập tin trên server
    - Tệp này có thể là bất kỳ loại tệp nào, như .txt và .xml hoặc các tệp tập lệnh máy chủ như .asp và .php (có thể thực hiện các hành động trên máy chủ trước khi gửi phản hồi lại).
  - *async*: true (bất đồng bộ) hay false (đồng bộ), mặc định là true
    - Bằng cách gửi không đồng bộ, JavaScript không phải đợi phản hồi của máy chủ
- VD:

```
xhttp.open("GET", "ajax_info.txt", true);
```

Sử dụng đối tượng XMLHttpRequest (tt)

4. Gửi yêu cầu đến máy chủ
   - send(): gửi yêu cầu đến server bằng phương thức GET
   - send(*string*): gửi yêu cầu đến server bằng phương thức POST
     - *string*: dữ liệu được gửi trong phần thân của thông điệp

VD:

```
const xhtml = new XMLHttpRequest();
xhtml.open("get", "example.php", false);
xhtml.send();
```

- Sau khi dữ liệu được gửi đi, và đối tượng nhận được thông điệp trả lời từ phía server, thì các thông tin về thông điệp trả lời này có thể được truy xuất thông qua các thuộc tính của đối tượng XMLHttpRequest

Sử dụng đối tượng XMLHttpRequest (tt)

- Các thuộc tính truy xuất đến thông tin của thông điệp trả lời:
  - *responseText* — nội dung (phần thân) của thông điệp trả lời
  - *responseXML* — chứa một XML DOM document nếu thông điệp trả về có kiểu nội dung (content type) là "text/xml" hay "application/xml"
  - *status* — mã trạng thái của thông điệp trả lời
  - *statusText* — chuỗi mô tả mã trạng thái
- *readyState* — xác định giai đoạn trong chu kỳ request/response đang được kích hoạt, các giá trị có thể có của thuộc tính này:
  - 0 — chưa được khởi tạo. Phương thức open() chưa được gọi
  - 1 — đã khởi tạo. Phương thức open() đã thực thi nhưng send() chưa được gọi
  - 2 — đã gửi. Phương thức send() đã được gọi, nhưng chưa nhận được thông điệp trả lời
  - 3 — đang nhận dữ liệu
  - 4 — hoàn tất. Đã nhận được tất cả dữ liệu trả về từ server

Sử dụng đối tượng XMLHttpRequest (tt)

- Khi thuộc tính readyState thay đổi giá trị, thì sự kiện onreadystatechange được phát sinh => cho phép viết các script kiểm tra giá trị của thuộc tính readyState, và xử lý dữ liệu gửi về từ server, *VD:*

```
var xhr=createXHR();
xhr.onreadystatechange=function(){
    if (xhr.readyState==4){
        if (xhr.status==200){
            alert(xhr.responseText)
        } else {
            alert("Request was unsuccessful: " + xhr.status);
        }
    }
};
xhr.open("get","example.txt",true);
xhr.send(null);
```

- Đối với các yêu cầu bất đồng bộ có thể được hủy bỏ bằng phương thức abort(), VD: xhr.abort();

HTTP Headers

- **setRequestHeader**("*header*", "*value*") — đặt các header field cho thông điệp yêu cầu của đối tượng XMLHttpRequest

  VD: xhr.setRequestHeader("MyHeader", "MyValue");

- **getResponseHeader**("*header*") — lấy về giá trị của một header field trong thông điệp trả lời

  VD: var headerValue=xhr.getResponseHeader("content-type");

- **getAllResponseHeaders**() — hàm trả về một chuỗi chứa tất cả các header field trong thông điệp trả lời

  VD: var allHeaders=xhr.getAllResponseHeaders()

  Kết quả của câu lệnh là biến allHeaders có giá trị là một chuỗi có định dạng giống như sau:

```
ETag: "500000003362e-36-4ad6ee0f34206"
Content-Length: 54
Keep-Alive: timeout=5, max=100
Content-Type: text/plain
Last-Modified: Wed, 21 Sep 2011 07:53:58 GMT
```

VD Gửi yêu cầu dùng chế độ bất đồng bộ

```
<html><body>
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>
<p id="demo"></p>
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("GET", "hello.php?fname=Henry&lname=Ford");
  xhttp.send();
}
</script>
</body></html>
```

VD Gửi yêu cầu dùng chế độ bất đồng bộ (tt)

- Nội dung file hello.php:

```php
<?php
echo "Hello ".$_REQUEST['fname']." ".$_REQUEST['lname']." !";
```

VD Gửi yêu cầu dùng chế độ đồng bộ

```
<html><body>
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>
<p id="demo"></p>
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.open("GET", "hello.php?fname=Henry&lname=Ford",false);
  xhttp.send();
  document.getElementById("demo").innerHTML = xhttp.responseText;
}
</script>
</body></html>
```

VD Gửi yêu cầu dùng chế độ đồng bộ (tt)

- Vì JavaScript sẽ chờ kết quả trả về server nên không cần tạo callback function ở chế độ đồng bộ

- Gửi yêu cầu ở chế độ đồng bộ không được khuyến khích sử dụng vì JavaScript sẽ ngừng thực thi cho đến khi phản hồi của máy chủ sẵn sàng. Nếu máy chủ bận hoặc chậm, ứng dụng sẽ bị treo hoặc dừng.

AJAX: GET or POST?

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

- Tuy nhiên, hãy luôn sử dụng các yêu cầu POST khi:
  - Không muốn trang yêu cầu bị cache
    - VD: Cập nhật tệp hoặc cơ sở dữ liệu trên máy chủ
  - Gửi một lượng lớn dữ liệu đến máy chủ
    - POST không có giới hạn về kích thước
  - Gửi dữ liệu người dùng nhập vào từ form (có thể chứa các ký tự không xác định)
    - POST mạnh mẽ và an toàn hơn GET

GET Requests

- Có thể gửi dữ liệu đến server bằng phương thức GET bằng cách nối chuỗi query-string vào cuối chuỗi URL
  - Cú pháp: <url>?name1=value1&name2=value2

- Các chuỗi name và value phải được mã hóa trước khi nối vào chuỗi URL bằng cách sử dụng hàm encodeURIComponent()

GET Requests (tt)

- **VD:**

```
function addURLParam(url, name, value){
    url +=(url.indexOf("?"))==-1 ? "?" : "&";
    url += encodeURIComponent(name)+"="+encodeURIComponent(value);
    return url;
}
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  let url = "hello.php";
  url = addURLParam(url, "fname", "A");
  url = addURLParam(url, "lname", "Nguyễn Văn");
  xhttp.open("GET", url);
  xhttp.send();
}
```

POST Requests

- Đối với phương thức POST để gửi dữ liệu về cho server ta truyền chuỗi query-string vào hàm send(), và thêm trường "Content-Type" có giá trị là "application/x-www-form-urlencoded" vào trong phần header của thông điệp yêu cầu

VD:

```
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("POST", "hello.php");
  xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xhttp.send("fname=Henry&lname=Ford");
}
```

POST Requests (tt)

- **Cách thức lấy dữ liệu từ HTML Form?**
  1. **Dùng FormData object**
     - Hầu hết các trình duyệt web hiện tại đều hỗ trợ FormData object
  2. **Tự viết hàm JavaScript** cho phép tạo chuỗi query string
     - Nếu ứng dụng cần hỗ trợ các phiên bản trình duyệt web cũ

1. Dùng FormData object

```
1  <html>
2  <body>
3  <script>
4  function loadDoc() {
5      const xhttp = new XMLHttpRequest();
6      xhttp.onload = function() {
7          document.getElementById("demo").innerHTML = this.responseText;
8          }
9      xhttp.open("POST", "hello.php");
10     const formData = new FormData(document.getElementById("inputForm"));
11     xhttp.send(formData);
12 }
13 </script>
14 <form id="inputForm">
15 <label for="fname">First Name: </label><input type="text" name="fname"> <br/>
16 <label for="lname">Last Name: </label><input type="text" name="lname"> <br/>
17 <button type="button" onclick="loadDoc()">Send data</button>
18 </form>
19 <h3>Message received from web server:</h3>
20 <p id="demo"></p>
21 </body>
22 </html>
```

2. Tự viết hàm JavaScript

```
function loadDoc() {
    const xhttp = new XMLHttpRequest();
    xhttp.onload = function() {
        document.getElementById("demo").innerHTML = this.responseText;
        }
    xhttp.open("POST", "hello.php");
    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    const formData = serialize(document.getElementById("inputForm"));
    console.log(formData);
    xhttp.send(formData);
}
```

2. Tự viết hàm JavaScript (tt)

```
1  function serialize(form){
2      let parts = new Array();
3      let field = null;
4      for (let i=0, len=form.elements.length; i  <  len; i++){
5          field = form.elements[i];
6          switch(field.type){
7          case "select-one":
8          case "select-multiple":
9              for (let j=0, optLen = field.options.length; j  <  optLen; j++){
10                 let option = field.options[j];
11                 if (option.selected){
12                     let optValue = "";
13                     if (option.hasAttribute){
14                         optValue = (option.hasAttribute("value") ?
15                                     option.value : option.text);
16                     } else {
17                         optValue = (option.attributes["value"].specified ?
18                                     option.value : option.text);
19                     }
20                     parts.push(encodeURIComponent(field.name) + "=" +
21                                encodeURIComponent(optValue));
22                 }
23             }
24             break;
```

2. Tự viết hàm JavaScript (tt)

```
25          case undefined:    //fieldset
26          case "file":        //file input
27          case "submit":      //submit button
28          case "reset":       //reset button
29          case "button":      //custom button
30              break;
31          case "radio":       //radio button
32          case "checkbox":    //checkbox
33              if (!field.checked){
34                  break;
35              }
36              /* falls through */
37          default:
38              parts.push(encodeURIComponent(field.name) + "=" +
39                  encodeURIComponent(field.value));
40          }
41      }
42      return parts.join(" & ");
43  }
```

4. AJAX & XML File

```
<script>
    function loadXMLFile(xmlFile){
        const xhttp = new XMLHttpRequest();
        xhttp.onload = myProcessingDataFunc;
        xhttp.open("GET", xmlFile);
        xhttp.send();
    }
    function myProcessingDataFunc(){
        const xmlDoc = this.responseXML;
        const x = xmlDoc.getElementsByTagName("CD");
        let tb = "<th>Artist</th><th>Title</th>";
        for( let i=0; i<x.length; i++){
            tb += "<tr><td>" + x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue + "</td>";
            tb += "<td>" + x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue + "</td></tr>";
        }
        document.getElementById("tbCatalog").innerHTML = tb;
    }
</script>
<body>
    <button onclick="loadXMLFile('cd_catalog.xml')">Load CD Catalog</button><br/><br/>
    <table id="tbCatalog"></table>
</body>
```

File cd_catalog.xml: https://www.w3schools.com/js/cd_catalog.xml

II. AJAX, PHP, & MySQL

II. AJAX, PHP, & MySQL

- AJAX:
  - Gửi yêu cầu/dữ liệu về web server được xử lý bằng PHP script
  - Nhận kết quả trả về từ web server

- PHP:
  - Nhận yêu cầu từ web client thông qua AJAX
  - Truy vấn cơ sở dữ liệu MySQL
  - Tạo kết quả trả về web client trong các định dạng sau:
    - Plain text/html (xem ví dụ 1)
    - JSON (xem ví dụ 2)
    - XML

Ví dụ 1

- Hiển thị danh sách khách hàng trong cơ sở dữ liệu classicmodels theo quốc gia

```php
<?php
    require_once("../mysqlConnect.php");
    $sqlTxt = "SELECT DISTINCT country FROM customers ORDER BY country";
    $rs = $mysqli->query($sqlTxt);
    $opts = "";
    while($row = $rs->fetch_assoc()){
        $opts .= "<option>".$row['country']."</option>";
    }
?>
<form id="frmCountry">
<Label>Load Customers from </label>
<select name="slCountry" id="slCountry"><?php echo $opts; ?></select>
</form>
<table id="tbCustomers"></table>
<script>
    document.getElementById("slCountry").onchange = function(){
        const xhttp = new XMLHttpRequest();
        xhttp.onload =function(){
            const txtHTML = this.responseText;
            document.getElementById("tbCustomers").innerHTML =txtHTML;
        }
        xhttp.open("POST", "../ajax_php/php_text.php");
        const formData = new FormData(document.getElementById("frmCountry"));
        xhttp.send(formData);
    };
</script>
```

Ví dụ 1 (tt)

- Nội dung file php_text.php:

```php
<?php
require_once("../mysqlConnect.php");
$country = $_REQUEST['slCountry'];

$stm = $mysqli->prepare('SELECT customerName, phone, city FROM customers WHERE country=?');
$stm->bind_param('s', $country);
$stm->execute();
$results = $stm->get_result();
$rs = "<table>";
$rs .= "<th>Name</th><th>Phone</th><th>City</th>";
while ($row = $results->fetch_assoc()) {
    $rs .= "<tr>";
    $rs .= "<td>".$row["customerName"]."</td>";
    $rs .= "<td>".$row["phone"]."</td>";
    $rs .= "<td>".$row["city"]."</td>";
    $rs .= "</tr>\n";
}
$rs .= "</table>";
echo $rs;
```

Ví dụ 2

- Nội dung file show_customers.php

```php
<?php
    require_once("../mysqlConnect.php");
    $sqlTxt = "SELECT DISTINCT country FROM customers ORDER BY country";
    $rs = $mysqli->query($sqlTxt);
    $opts = "";
    while($row = $rs->fetch_assoc()){
        $opts .= "<option>".$row['country']."</option>";
    }
?>
<form id="frmCountry">
<Label>Load Customers from </label>
<select name="slCountry" id="slCountry"><?php echo $opts; ?></select>
</form>
<table id="tbCustomers"></table>
```

Ví dụ 2 (tt)

- Nội dung file show_customers.php

```
<script>
    document.getElementById("slCountry").onchange = function(){
        const xhttp = new XMLHttpRequest();
        xhttp.onload = processJsonDataFunc;
        xhttp.open("POST", "../ajax_php/php_json.php");
        const formData = new FormData(document.getElementById("frmCountry"));
        xhttp.send(formData);
    };
    function processJsonDataFunc(){
        js = JSON.parse(this.responseText);
        let tb = "<th>Name</th><th>Phone</th><th>City</th>";
        for( let i=0; i<js.length; i++){
            row = js[i];
            tb += "<tr><td>" + row["customerName"] + "</td>";
            tb += "<td>" + row["phone"]+ "</td>";
            tb += "<td>" + row["city"] + "</td></tr>";
        }
        document.getElementById("tbCustomers").innerHTML = tb;
    }
</script>
```

Ví dụ 2 (tt)

- Nội dung file php_json.php

```php
<?php
header("Content-Type: application/json; charset=UTF-8");
require_once("../mysqlConnect.php");
$country = $_REQUEST['slCountry'];
$stm = $mysqli->prepare('SELECT customerName, phone, city
                         FROM customers WHERE country=?');
$stm->bind_param('s', $country);
$stm->execute();
$results = $stm->get_result();
$outp = $results->fetch_all(MYSQLI_ASSOC);
echo json_encode($outp);
```

Question ?

THANK YOU !