Curs TWSS

Utilizare Ajax și PHP

- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML
  - Set de metode Javascript prin intermediul cărora se transferă date între browser și serverul web
  - Tehnică prin intermediul căreia se creează aplicații web interactive

Crearea unui XmlHttpRequest

```
<script>
function ajaxRequest()
{
try // Non IE Browser?
 var request = new XMLHttpRequest()
catch(e1)
{
 try // IE 6+?
  request = new ActiveXObject("Msxml2.XMLHTTP")
 catch(e2)
 {
 try // IE 5?
  request = new ActiveXObject("Microsoft.XMLHTTP")
 catch(e3) // There is no Ajax Support
  request = false
 }
}
return request
}
</script>
```

- Metode:
  - **abort()** - Aborts the current request.
  - **getAllResponseHeaders()** - Returns all headers as a string.
  - **getResponseHeader(param)** - Returns the value of param as a string.
  - **open('method', 'url', 'asynch')** - Specifies the HTTP method to use (GET or POST), the target URL, and whether the request should be handled asynchronously (true or false).
  - **send(data)** - Sends data to the target server using the specified HTTP method.
  - **setRequestHeader('param', 'value')** - Sets a header with a parameter/value pair.

- Proprietăți:
  - **onreadystatechange** - Specifies an event handling function to be called whenever the *readyState* property of an object changes.
  - **readyState** – un întreg care raportează care e starea apelului Ajax. Valorile posibile sunt: 0 = Uninitialized, 1 = Loading, 2 = Loaded, 3 = Interactive, and 4 = Completed.
  - **responseText** - The data returned by the server in text format.

- **responseXML** - The data returned by the server in XML format.
- **status** - The HTTP status code returned by the server.
- **statusText** - The HTTP status text returned by the server.

```
<div id='info'>This sentence will be replaced</div>
<script>
 request = new ajaxRequest()
 request.open("POST", "urlpost.php", true)
 request.setRequestHeader("Content-type", "application/x-www-form-
urlencoded")
 request.setRequestHeader("Content-length", params.length)
 request.setRequestHeader("Connection", "close")
 request.onreadystatechange = function(){
  if (this.readyState == 4)
   if (this.status == 200)
    if (this.responseText != null)
     document.getElementById('info').innerHTML = this.responseText
    else
     alert("Ajax error: No data received")
   else
    alert( "Ajax error: " + this.statusText)
 }
 request.send("url=oreilly.com")
 function ajaxRequest(){
 try
  var request = new XMLHttpRequest()
 catch(e1)
  {
  try
   request = new ActiveXObject("Msxml2.XMLHTTP")
  catch(e2)
   {
   try
    request = new ActiveXObject("Microsoft.XMLHTTP")
   catch(e3)
    request = false
   }
  }
 return request
 }
</script>
```

```
<?php
 if (isset($_POST['url'])) {
  echo file_get_contents("http://".SanitizeString($_POST['url']));
 }
 function SanitizeString($var) {
  $var = strip_tags($var);
  $var = htmlentities($var);
  return stripslashes($var);
 }
```

```
?>
```

```
<div id='info'>This sentence will be replaced</div>
<script>
 nocache = "&nocache=" + Math.random() * 1000000
 request = new ajaxRequest()
 request.open("GET", "urlget.php?url=oreilly.com" + nocache, true)
 request.onreadystatechange = function(){
  if (this.readyState == 4)
   if (this.status == 200)
    if (this.responseText != null)
     document.getElementById('info').innerHTML = this.responseText
    else
     alert("Ajax error: No data received")
   else
    alert( "Ajax error: " + this.statusText)
 }
 request.send(null)
 function ajaxRequest(){
 try
  var request = new XMLHttpRequest()
 catch(e1)
  {
  try
   request = new ActiveXObject("Msxml2.XMLHTTP")
  catch(e2)
   {
   try
    request = new ActiveXObject("Microsoft.XMLHTTP")
   catch(e3)
    request = false
   }
  }
 return request
 }
</script>
```

```
<?php
 if (isset($_GET ['url'])) {
  echo file_get_contents("http://".SanitizeString($_GET['url']));
 }
 function SanitizeString($var) {
  $var = strip_tags($var);
  $var = htmlentities($var);
  return stripslashes($var);
 }
?>
```

PHP & AJAX - JSON

- JSON = **J**ava**S**cript **O**bject **N**otation
- Datele trimise între browser și serverul web sunt în format text
- Obiectele JS se pot converti în JSON și reciproc
- Format:
  - Informațiile sunt stocate sub forma unor perechi (nume, valoare)
  - În formatul JSON
    - **Cheile** trebuie să fie șiruri de caractere încapsulate între apostroafe duble
    - **Valorile** trebuie să fie sir de caractere, număr, obiect JSON, șir, boolean, null
  - Perechile sunt separate prin virgule
  - Acoladele separă obiectele
  - Parantezele drepte separă șirurile
- Utilizare:
  - Apelul AJAX returnează o listă de obiecte;
  - Prin evaluarea răspunsului se obține un obiect JS care reprezintă un vector asociativ, prin intermediul căruia fiecare element are mapate atributele corespunzătoare
  - Astfel, nu mai este necesară parsarea răspunsului

Data JSON (nume,valoare)	{ "name":"John" }
Obiect JSON	{ "name":"John", "age":31, "city":"New York" };
	{"employees":[{ "firstName":"John", "lastName":"Doe" }, { "firstName":"Anna", "lastName":"Smith" }, { "firstName":"Peter", "lastName":"Jones" }]}
Accesarea valorii unui obiect JSON	myObj = { "name":"John", "age":30, "car":null }; x = myObj.name;
	myObj = { "name":"John", "age":30, "car":null }; x = myObj["name"];

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
      var myObj = JSON.parse(this.responseText);
      document.getElementById("demo").innerHTML = myObj.name;
   }
};
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
```

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
      var myArr = JSON.parse(this.responseText);
      document.getElementById("demo").innerHTML = myArr[0];
   }
};
xmlhttp.open("GET", "json_demo_array.txt", true);
xmlhttp.send();
```

```php
<?php
$myObj->name = "John";
$myObj->age = 30;
$myObj->city = "New York";
$myJSON = json_encode($myObj);
echo $myJSON;
?>
```

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myObj = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myObj.name;
    }
};
xmlhttp.open("GET", "demo_file.php", true);
xmlhttp.send();
```

HTACCESS

- .htaccess
  - Reguli de rescriere
  - Reguli de redirecționare
- Regulile scrise în .htaccess au efect asupra fișierelor din folder-ul în care acesta se află și asupra tuturor subdirectoarelor

- Reguli de rescriere (rewrite)

RewriteEngine On	Permite utilizarea regulilor de rewriting
RewriteRule ^about$ about.html	Redirecționează utilizatorii către about.html dacă navighează către /about
RewriteRule ^([0-9a-z]+)$ $1.html	Redirecționează utilizatorii către requestedpath.html dacă navighează către /requestedpath
RewriteCond %{REQUEST_FILENAME} !-d	Se verifică dacă prin cererea făcută nu se navighează către un director sau fișier existent. Acestea trebuie să fie scrise înaintea regulilor de rescriere care dorim să respecte condiția
RewriteCond %{REQUEST_FILENAME} !-f	

- Reguli de rescriere (rewrite) – Flag-uri
  - Flag-urile se scriu la finalul fiecarei reguli de rescriere pentru a specifica informații adiționale

QSA	Adaugă query string-ul care a fost atașat cererii: /about?foo=bar will become /about.php?foo=bar.
RewriteRule ^([0-9a-z]+)$ index.php?page=$1 [QSA]	/about -> index.php?page=about /bacon?foo=bar -> index.php?page=bacon&foo=bar

- *Reguli de redirecționare – folosirea flag-ului R*

QSA	Adaugă query string-ul care a fost atașat cererii: /about?foo=bar will become /about.php?foo=bar.
RewriteRule ^([0-9a-z]+)$ index.php?page=$1 [QSA]	/about -> index.php?page=about /bacon?foo=bar -> index.php?page=bacon&foo=bar

- *Specificarea paginilor de eroare*

ErrorDocument xxx /xxx	ErrorDocument 404 /errors/notfound.html

- Specificarea paginilor de index

DirectoryIndex maintenance.html new_index.html index.html

Exemplu

```
ErrorDocument 404 /404
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^rssfeed\.rss$ /custom/feed.php [L]
RewriteRule ^([0-9a-zA-Z\_\-]+)\/$ $1 [R=301,L]
RewriteRule ^([0-9a-zA-Z\_\-]+)$ page.php?id=$1 [QSA]
```

CSRF

- Cross Site Request Forgery
- Poate fi folosit pentru a pacali un utilizator autentificat sa faca anumite modificari neintentionate pe site
- Profita de increderea pe care o ofera site-ul utilizatorilor deja autentificati
- Un atac CSRF este de succes daca utilizatorii sunt identificati cu ajutorul cookie-urilor si sunt autentificati in timpul atacului
- Modalitati:
    - Link pe care utilizatorul sa dea click
    - Includerea unor mecanisme care solicita resursa fara a fi nevoie ca utilizatorul sa dea click

- **Scenariu**
    - Boby navigheaza pe ***mysite.com***, si este autentificat ca si utilizator legitim.
    - ***Mysite.com*** are un buton, care atunci cand este apasat, va trimite automat un e-card la toate contactele lui Boby salvate, automat cu un link.
    - In timp ce Boby navigheaza, are o intrebare despre anumite functii, si merge pe forumul favorit cu informatii despre ***mysite.com*** – mysiteforums.com
    - El deschide un articol interesant al unui alt utilizator. Acesta contine o eticheta cu imagine care arata in felul urmator:
    - Browserul lui Boby incearca sa incarce imaginea, si trimite datele de autentificare ale lui Bob impreuna cu solicitarea pentru mysite.com/send-e-cards.
    - ***Mysite.com*** primeste cererea de la Bob pentru trimiterea e-card-ului, si o trimite.
    - Atacul a fost realizat cu succes, chiar daca Bob nu a apasat niciodata butonul send.
    - Acest tip de atac a fost utilizat cu succes in raspandirea virusilor, atat prin intermediul MySpace, cat si Facebook.
    - Atacurile de aceasta natura, faciliteaza autopropagarea, deoarece ataca site-uri unde utilizatorii isi stocheaza informatii personale.

- **Scenariu GET**
    - Alice doreste sa ii transfere lui Bobo 100$ folosind aplicatia bank.com. Maria (atacatorul) doreste sa o pacaleasca pe Alica sa ii trimita banii ei.
    - GET http://bank.com/transfer.do?acct=BOB&amount=100 HTTP/1.1
    - Maria decide sa exploateze vulnearabilitatea aplicatie si sa o foloseasca pe Alice drept victim
    - http://bank.com/transfer.do?acct=MARIA&amount=100000
    - Aspectul social (pacalirea lui Alice)
        - Maria trimite un e-mail cu continut HTML
        - Maria planteaza un URL sau un script intr-o pagina care e posibil sa fie vizitata de victim
        - View my Pictures!
        - <img src="http://bank.com/transfer.do?acct=MARIA&amount=100000" width="0" height="0" border="0">
- Daca aceasta imagine ar fi inclusa intr-un e-mail, Alice nu ar vedea nimic. Totusi, browser-ul va trimite cererea catre bank.com facand posibil transferul

- **Scenariu POST**
  - POST http://bank.com/transfer.do HTTP/1.1
    acct=BOB&amount=100
  - <form action="http://bank.com/transfer.do" method="POST">
    <input type="hidden" name="acct" value="MARIA"/>
    <input type="hidden" name="amount" value="100000"/>
    <input type="submit" value="View my pictures"/>
    </form>
  - <body onload="document.forms[0].submit()">
    <form...

- **Scenariu folosind alte metode HTTP**
  - PUT http://bank.com/transfer.do HTTP/1.1
    { "acct":"BOB", "amount":100 }
  - <script>
    function put() {
    var x = new XMLHttpRequest();
    x.open("PUT","http://bank.com/transfer.do",true);
    x.setRequestHeader("Content-Type", "application/json");
    x.send(JSON.stringify({"acct":"BOB", "amount":100}));
    }
    </script>
    <body onload="put()">

Cuprins