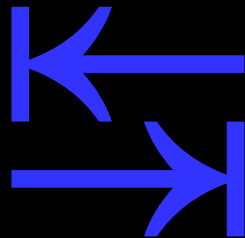


# Tehnologii Web

## interacțiune Web



suita de tehnologii **Ajax**  
aplicații Web hibride (*mash-ups*)

“Modul în care dăm  
face mai mult decât ceea ce dăm.”

**Pierre Corneille**

Care e modalitatea de a transfera asincron  
date între client(i) și server(e) Web?

# interacțiune web: ajax

*Asynchronous JavaScript And XML*  
(Jeese James Garrett)

# interacțiune web: ajax

## *Asynchronous JavaScript And XML*

(Jeese James Garrett)

permite transfer asincron de date  
între un document HTML redat de client (*browser*)  
și o aplicație rulând pe un server Web

# interacțiune web: ajax

## *Asynchronous JavaScript And XML*

(Jeese James Garrett)

oferă premisele asigurării unei interacțiuni complexe  
cu utilizatorul (RIA – *Rich Internet Application*)  
în cadrul unei aplicații Web

# interacțiune web: ajax

O suită de tehnologii deschise:

limbaje standardizate de structurare – *e.g.*, (X)HTML –  
și de prezentare a datelor: CSS

# interacțiune web: ajax

O suită de tehnologii deschise:

redare + interacțiune la nivel de client (navigator) Web  
via standardul DOM



# interacțiune web: ajax

O suită de tehnologii deschise:

interschimb + prelucrare de date reprezentate prin:  
diverse dialecte XML,  
JSON (*JavaScript Object Notation*),  
HTML,  
alte formate

# interacțiune web: **ajax**

O suită de tehnologii deschise:

transfer (a)sincron de date via HTTP  
facilitat de obiectul XMLHttpRequest

# interacțiune web: ajax

O suită de tehnologii deschise:

procesare folosind limbajul JavaScript (ECMAScript)

# interacțiune web: ajax

Componenta de bază: obiectul **XMLHttpRequest**

disponibil la nivelul navigatorului Web via JavaScript

# interacțiune web: ajax

Componenta de bază: obiectul **XMLHttpRequest**

nivelul 1 (în mod normal, implementat de orice *browser*)

[www.w3.org/TR/XMLHttpRequest1/](http://www.w3.org/TR/XMLHttpRequest1/)

nivelul 2 (pentru navigatoare recente) – în lucru la W3C

[www.w3.org/TR/XMLHttpRequest/](http://www.w3.org/TR/XMLHttpRequest/)

# interacțiune web: **ajax**

Componenta de bază: obiectul **XMLHttpRequest**

permite realizarea de cereri HTTP – *e.g.*, GET, POST,... –  
dintr-un program rulând la nivel de client (*browser*)  
spre o aplicație / un serviciu Web existent(ă) pe server,  
în mod **asincron** ori **sincron**

# interacțiune web: ajax

Componenta de bază: obiectul **XMLHttpRequest**

datele vehiculate între programele client și server  
pot avea orice format

uzual, modelate în XML (*e.g.*, Atom, RSS, KML,...),  
HTML și/sau JSON

# interacțiune web: ajax

Componenta de bază: obiectul **XMLHttpRequest**

paginile Web nu mai trebuie reîncărcate complet,  
conținutul lor – structurat via HTML –  
fiind manipulat prin DOM în cadrul *browser*-ului,  
în conformitate cu datele recepționate de la server



```
interface XMLHttpRequest : XMLHttpRequestEventTarget {  
  // funcția de tratare a evenimentului de schimbare a stării transferului  
  attribute Function? onreadystatechange;  
  readonly attribute unsigned short readyState; // starea transferului  
  
  // realizarea unei cereri HTTP  
  // deschide o conexiune cu serverul Web  
  void open (ByteString metoda, [EnsureUTF16] DOMString url);  
  void open (ByteString metoda, [EnsureUTF16] DOMString url,  
              boolean async, optional ByteString? numecont = null,  
              optional DOMString? parola = null);  
  
  // stabilește antetul HTTP  
  void setRequestHeader (ByteString campAntet, ByteString valoare);  
  // stochează valoarea în milisecunde a timpului maxim de așteptare  
  attribute unsigned long timeout;  
  
  void send (optional data = null); // trimite date spre serverul Web  
  void abort (); // abandonează transferul
```

```
// receptarea răspunsului de la serverul Web
// codul de stare HTTP emis de server: 200, 303, 400,...
readonly attribute unsigned short status;
// textul asociat codului de stare
readonly attribute ByteString statusText;
```

```
// preia valoarea câmpului-antet din mesajul HTTP transmis de server
ByteString? getResponseHeader (ByteString antet);
// furnizează toate câmpurile răspunsului
ByteString getAllResponseHeaders ();
```

```
// specifică tipul MIME al răspunsului: blob, document, json, text,...
attribute XMLHttpRequestResponseType responseType;
// conține răspunsul propriu-zis
readonly attribute any response;
// furnizează răspunsul în format text
readonly attribute DOMString responseText;
// stochează răspunsul ca document XML
readonly attribute Document? responseXML;
};
```

```
interface XMLHttpRequestEventTarget : EventTarget {  
  // funcții de tratare a evenimentelor asociate transferului asincron  
  attribute Function? onloadstart; // transferul a început  
  attribute Function? onprogress; // se realizează transferul datelor...  
  attribute Function? onabort;     // s-a abandonat transferul de date  
  attribute Function? onerror;     // a apărut o eroare de transmisie  
  attribute Function? onload;      // datele au fost recepționate de client  
  attribute Function? ontimeout;   // a apărut o întârziere de transfer  
  attribute Function? onloadend;   // transferul s-a terminat  
};  
  
// constante ce specifică starea transferului (vezi proprietatea readyState)  
const unsigned short UNSENT = 0; // încă n-au fost efectuate transferuri  
const unsigned short OPENED = 1; // s-a deschis conexiunea cu serverul  
const unsigned short HEADERS_RECEIVED = 2; // primire câmpuri-antet HTTP  
const unsigned short LOADING = 3; // datele propriu-zise se încarcă  
const unsigned short DONE = 4;   // gata! (transfer efectuat complet)
```

vezi cursul  
despre DOM

specificație WebIDL

# interacțiune web: ajax

Metode importante oferite de **XMLHttpRequest**

**open ( )**

inițiază – deschide – o conexiune HTTP cu serverul,  
emițând o cerere: GET, POST,...

# interacțiune web: ajax

Metode importante oferite de XMLHttpRequest

**send ( )**

transmite (asincron) date – *e.g.*, XML, JSON etc. –,  
spre aplicația/serviciul ce rulează pe server

# interacțiune web: ajax

Metode importante oferite de XMLHttpRequest

**send ( )**

transmite (asincron) date – *e.g.*, XML, JSON etc. –,  
spre aplicația/serviciul ce rulează pe server

orice *listener* (asociat evenimentelor onload, ontimeout, onabort,...) trebuie stabilit înainte de a trimite date

# interacțiune web: ajax

Metode importante oferite de XMLHttpRequest

**abort ( )**

abandonează transferul de date curent

# interacțiune web: ajax

Metode importante oferite de **XMLHttpRequest**

**setRequestHeader ( )**

specifică anumite câmpuri de antet HTTP

exemple: **Cookie, Keep-Alive, User-Agent,...**



# interacțiune web: ajax

Metode importante oferite de **XMLHttpRequest**

**getResponseHeader ( )**

furnizează un anumit câmp prezent  
în antetul mesajului de răspuns HTTP trimis de server

# interacțiune web: ajax

Metode importante oferite de **XMLHttpRequest**

**getAllResponseHeaders ( )**

oferă toate câmpurile HTTP trimise de server,  
exceptând **Set-Cookie**

# interacțiune web: **ajax**

Proprietăți de bază ale **XMLHttpRequest**

**readyState**

furnizează codul de stare a transferului:

0 – **UNSENT**, 1 – **OPENED**,  
2 – **HEADERS\_RECEIVED**, 3 – **LOADING**, 4 – **DONE**

# interacțiune web: ajax

Proprietăți de bază ale XMLHttpRequest

status

oferă codul de stare HTTP întors de serverul Web:

200 (*Ok*)

404 (*Not Found*)

500 (*Internal Server Error*)

...

# interacțiune web: **ajax**

Proprietăți de bază ale **XMLHttpRequest**

**statusText**

conține mesajul corespunzător codului de stare HTTP

# interacțiune web: **ajax**

Proprietăți de bază ale **XMLHttpRequest**

**responseText**  
**responseXML**

stochează răspunsul (datele) obținut(e) de la server

# interacțiune web: **ajax**

Proprietăți de bază ale **XMLHttpRequest**

**onreadystatechange**

specifică funcția ce va fi invocată la modificările de stare ale transferului de date dintre server și client

*handler* de tratare a  
evenimentelor de transfer

# interacțiune web: **ajax**

Excepții ce pot fi emise

**AbortError**  
**InvalidAccessError**  
**InvalidStateError**  
**NetworkError**  
**SecurityError**  
**TimeoutError**

...

conform DOM 4 Core



Ce alte aspecte trebuie considerate  
atunci când se recurge la Ajax?

# interacțiune web: ajax – utilizări

Reîmprospătarea periodică a conținutului

*e.g.*, știri recepționate în formate ca Atom sau RSS,  
mesaje în cadrul aplicațiilor sociale, notificări,...

# interacțiune web: ajax – utilizări

Anticiparea *download*-urilor

pre-încărcarea datelor (*e.g.*, imagini) ce vor fi solicitate

# interacțiune web: ajax – utilizări

Auto-completarea datelor

*auto-completion*

sugestii de căutare – exemplu: Google Suggest

# interacțiuni web: ajax – utilizări

Validarea în timp-real a datelor introduse  
în formulare de către utilizator

exemplificare:

verificarea existenței unui cont sau a unei localități

# interacțiune web: ajax – utilizări

Creare de componente de interfață Web (*widgets*)  
sau de aplicații Web rulând pe platforme mobile

interacționează cu utilizatorul  
pe baza evenimentelor survenite

# interacțiune web: ajax – aspecte

Evitarea încărcării întregului document Web

avantaj:

se pot modifica doar fragmente de document

dezavantaj:

*bookmarking*-ul poate fi compromis  
(nu există un URL unic desemnând  
reprezentarea resursei curente)

# interacțiune web: ajax – aspecte

Oferirea de alternative la Ajax,  
atunci când suportul pentru acesta  
nu este implementat/activat

*graceful degradation*

*progressive enhancement*



# interacțiune web: ajax – aspecte

Minimizarea traficului dintre *browser* și server

transferul de date poate fi monitorizat (+interceptat)  
via instrumente dedicate

WireShark

Firebug, Fiddler, TamperData, Live HTTP Headers

# interacțiune web: ajax – aspecte

Stabilirea unui mod clar de interacțiune

interacțiune HTML clasică

*versus*

interacțiune „bogată” cu Ajax

*versus*

interacțiune la nivelul unei aplicații convenționale

# interacțiune web: ajax – aspecte

Adoptarea Ajax pentru creșterea utilizabilității,  
nu doar de dragul tehnologiei

exemple negative:

distragerea utilizatorului

abuz de resurse (supradimensionarea arborelui DOM)

# interacțiune web: ajax

Ajax oferă premisele invocării asincrone de servicii Web în stilul REST

folosind ca reprezentări ale datelor transferate:

*POX (Plain Old XML)*

*JSON (JavaScript Object Notation)*

*AHAH (Asynchronous HTML and HTTP)*

text neformatat

Care e suportul vizând implementarea?

# interacțiune web: ajax – programare

La nivel de client  
(biblioteci + *framework*-uri JavaScript)

Dojo: [dojotoolkit.org](http://dojotoolkit.org)

jQuery: [jquery.com](http://jquery.com)

Prototype: [prototypejs.org](http://prototypejs.org)

Rico: [openrico.org](http://openrico.org)

Script.aculo.us: [script.aculo.us](http://script.aculo.us)

alte: <http://www.javascripting.com/search?q=ajax>

# interacțiune web: ajax – programare

La nivel de server

biblioteci, module, *framework*-uri

Apache Wicket, DWR, Vaadin etc. (Java)

Ajax Control Toolkit, MagicAjax.NET (.NET)

Express, nCombo, socket.io, Tower etc. (Node.js)

Cjax, Sajax, Symfony, Yii,... (PHP)

CGI::Ajax, Catalyst, Mason (Perl)

Ruby on Rails (Ruby)

...

# interacțiune web: ajax – programare

API-uri specializate

exemplificări:

Bing Maps AJAX Control

<http://msdn.microsoft.com/en-us/library/gg427610.aspx>

Nokia HERE

<https://developer.here.com/javascript-apis>

Ajax în contextul extensiilor WordPress

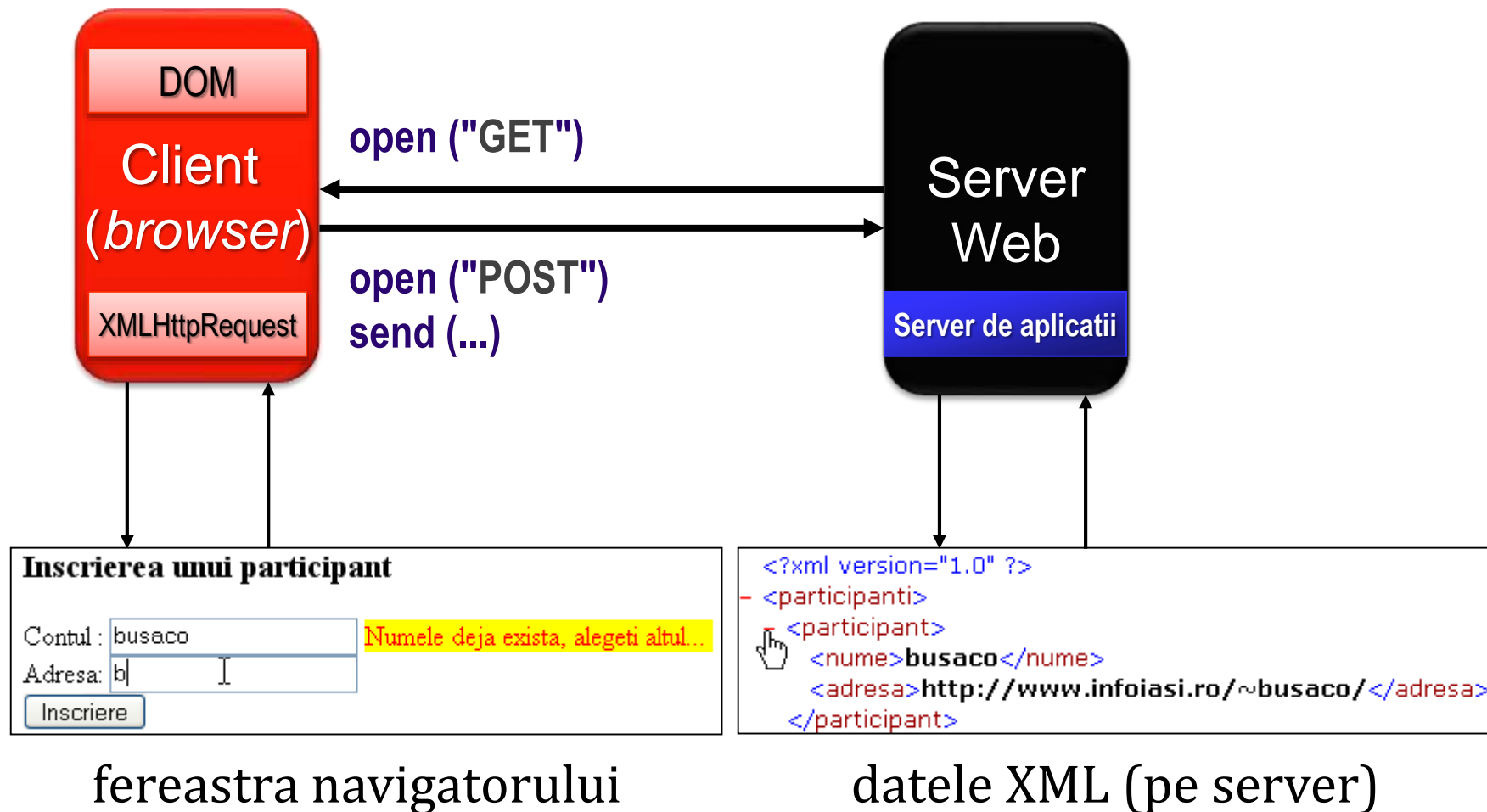
[http://codex.wordpress.org/AJAX\\_in\\_Plugins](http://codex.wordpress.org/AJAX_in_Plugins)



# interacțiune web: **ajax** – studiu de caz

Verificarea existenței unui nume de utilizator  
în vederea creării unui cont de autentificare  
în cadrul unei aplicații Web

# interacțiune web: ajax – exemplu



verificarea existenței unui cont pe server

# interacțiune web: **ajax** – studiu de caz

Verificarea existenței unui nume de utilizator  
în vederea creării unui cont de autentificare  
în cadrul unei aplicații Web

tratând prin DOM evenimentul **onblur**, putem detecta  
– interogând asincron aplicația Web de pe server –  
faptul că numele de cont introdus de utilizator  
într-un formular Web deja a fost folosit de altcineva

# interacțiune web: **ajax** – studiu de caz


Verificarea existenței unui nume de utilizator  
în vederea creării unui cont de autentificare  
în cadrul unei aplicații Web

aplicația Web de pe server – adoptând stilul REST –  
va oferi un document XML modelând răspunsul la  
interogarea „există deja un utilizator având un nume dat?”

```
var cerere; // încapsulează cererea HTTP către serverul Web
```

```
function incarcaXML (url) { // încarcă un document XML desemnat de 'url'  
    // verificăm existența obiectului XMLHttpRequest  
    if (window.XMLHttpRequest) {  
        cerere = new XMLHttpRequest (); // există suport nativ  
    } else  
        if (window.ActiveXObject) { // se poate folosi obiectul ActiveX din MSIE  
            cerere = new ActiveXObject ("Microsoft.XMLHTTP");  
        }  
    if (cerere) { // există suport pentru Ajax  
        // stabilim funcția de tratare a stării transferului de date  
        cerere.onreadystatechange = trateazaRaspunsCerere;  
        // preluăm documentul prin metoda GET  
        cerere.open ("GET", url, true);  
        cerere.send (null); // nu trimitem nimic serviciului Web  
    }  
}
```

```
// funcția de tratare a schimbării de stare a cererii
function trateazaRaspunsCerere () {
    // verificăm dacă încărcarea s-a terminat cu succes
    if (cerere.readyState == 4) {
        // am obținut codul de stare '200 Ok'?
        if (cerere.status == 200) {
            // procesăm datele recepționate prin DOM
            // (preluăm elementul rădăcină al documentului XML)
            var raspuns = cerere.responseXML.documentElement;
            var rezultat = raspuns.getElementsByTagName
                ('rezultat')[0].firstChild.data;
            // apelăm o funcție ce va modifica arborele DOM al paginii Web
            // conform răspunsului transmis de serviciul invocat
            ...
        }
        // eventual, se pot trata și alte coduri HTTP (404, 500 etc.)
    } else {
        alert ("Problemă la transferul datelor XML:\n" + cerere.statusText);
    }
}
```



vezi  
exemplul  
din arhivă

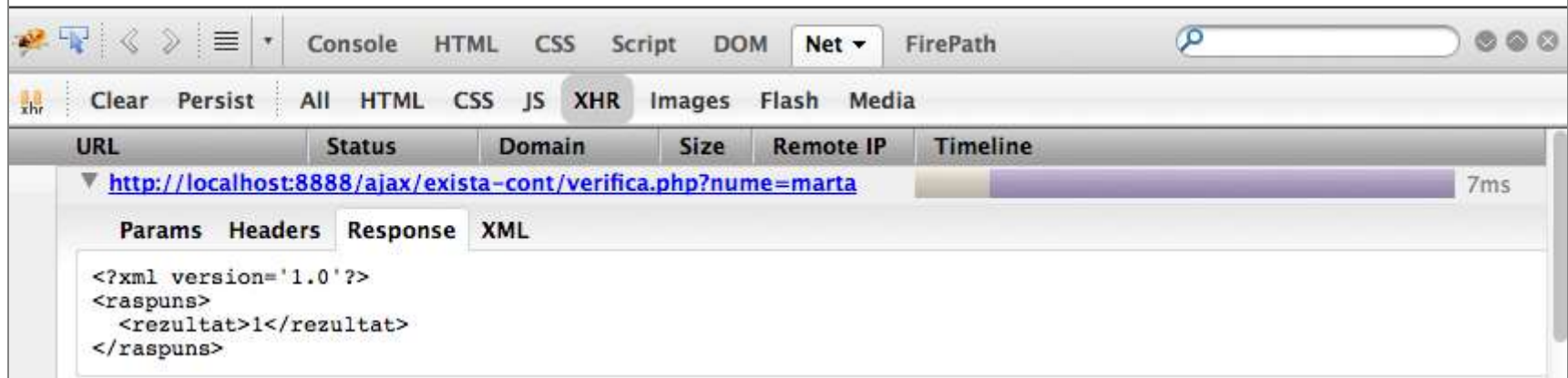
# Inscrierea unui participant

Numele contului:

Numele deja exista, alegeti altul...

Adresa d-voastra:

Inscriere



utilizatorul introduce un nume de cont; via Ajax, i se va semnala că deja există, conform răspunsului XML trimis de către serviciul Web

cerere HTTP via URL-ul <http://adresa.info/verifica.php?nume=marta>  
răspuns XML de forma `<raspuns><rezultat>1</rezultat></raspuns>`

0 = nu există

# studiu de caz: FlickrPics

Obținerea fotografiilor publice stocate pe situl  
Flickr pe baza serviciului Web oferit

cod-sursă disponibil la  
<http://jsfiddle.net/busaco/4d2tmc6b/>



# studiu de caz: FlickrPics

Obținerea fotografiilor publice stocate pe situl Flickr pe baza serviciului Web oferit

utilizăm URL-ul

[http://api.flickr.com/services/feeds/photos\\_public.gne](http://api.flickr.com/services/feeds/photos_public.gne)

pentru a obține informații despre imagini  
(formate disponibile: Atom, CSV, JSON, XML,...)

vezi [http://www.flickr.com/services/feeds/docs/photos\\_public/](http://www.flickr.com/services/feeds/docs/photos_public/)

Forma generală a răspunsului JSON transmis de Flickr:

```
{
  "title"      : "Recent Uploads",
  "link"       : "http://www.flickr.com/photos/",
  "modified"   : "2015-05-19T09:52:07Z",
  "generator"  : "http://www.flickr.com/",
  "items"     : [ {
    "title"    : "...",
    "link"     : "http://www.flickr.com/photos/.../4204222/",
    "media" : { "m": "https://farm.staticflickr.com/...jpg" },
    "date_taken": "2012-05-20T17:23:43-08:00",
    "description": "...",
    "published"  : "2012-05-26T13:49:08Z",
    "author"     : "...",
    "author_id"  : "...",
    "tags"       : "iasi romania informatica FII ..."
  } ]
}
```

# studiu de caz: FlickrPics

```
// preluăm asincron imagini disponibile pe Flickr
jQuery.getJSON
("http://api.flickr.com/services/feeds/photos_public.gne?jsoncallback=?",
{ // datele de intrare transmise serviciului Web
  tags: "lasi, informatica", format: "json"
},
// funcția anonimă ce va procesa datele JSON trimise asincron de Flickr
function (data) {
  // iterăm fiecare informație obținută de la serviciul Web
  $.each (data.items, function (numar, foto) {
    // creăm un element <img> având ca valoare a atributului "src"
    // adresa Web inclusă în datele JSON obținute;
    // acest <img> va fi adăugat la elementul cu id="imagini" din pagină
    $("<img/>").attr("src", foto.media.m).attr("title", foto.title)
      .appendTo("#imagini");
  });
});
```

# studiu de caz: FlickrPics

JSFIDDLE

Run Update Fork TidyUp JSHint Collaboration Share

Login/Sign up

Frameworks & Extensions

jQuery 1.8.3

☐ jQuery Mobile 1.2.0

☐ jQuery UI 1.9.2

No wrap - in <head>

Fiddle Options

External Resources

Languages

Ajax Requests

Legal, Credits and Links

```
1 <article>
2   <div id="imagini">
3     <!-- in cadrul acestui element vor fi incluse imaginile
4   -->
5   </div>
6 </article>
```

```
26 $.getJSON ("http://api.flickr.com/services/feeds
27 /photos_public.gne?jsoncallback=?", { // datele de intrare
28   trimise serviciului Web
29   tags: "Iasi, informatica",
30   tagmode: "all",
31   format: "json" // dorim JSON (formatul implicit este Atom)
32 },
33 // functia anonima care va procesa datele JSON
34 // transmite asincron de catre Flickr
35 function (data) {
36   // iteram fiecare informatie obtinuta de la serviciul web
37   $.each(data.items, function (numar, foto) {
38     // 'iesim' din iterator daca am depasit maximul dorit
39     if (numar >= MAX_IMG) return false;
40     // cream un element <img> avand ca valoare a
41     atributului "src"
42     // adresa Web inclusa in datele JSON obtinute;
43     // acest <img> va fi adaugat la elementul cu
44     id="imagini"
45     $("<img/>")
46       .attr("src", foto.media.m)
47       .attr("title", foto.title)
48       .appendTo("#imagini");
```

```
1 img {
2   width: 300px;
3   padding: 0.2em;
4   margin: 0.2em;
5   float: left;
6   border: thin solid #ccc;
7 }
```

Preview images:

- A person presenting to an audience.
- A hand holding a document with a diagram.

un posibil rezultat – editarea & rularea codului via JSFiddle

# interacțiune web: **ajax** – studiu de caz

Generalizând, putem recurge la metoda **ajax ()**:

```
jQuery.ajax ({ // execută o cerere POST pentru invocarea serviciului Web
  type: "POST",
  contentType: "application/json; charset=utf-8",
  url: "http://undeva.info/ServiciuWeb/Resursa",
  data: "{...}",           // datele de intrare trimise serviciului
  dataType: "json",        // așteptăm răspunsul în format JSON
  success: function (data) { // funcție apelată la transferul cu succes
    $('.rezultat').html (data); // preluăm datele, convertindu-le în HTML
  }
});
```



# ajax: demo





# interacțiune web: comet

## *Comet*

termen propus de Alex Russel (2006)

permite ca datele să fie „împinse” (*push*) de către server spre aplicația client, utilizând conexiuni HTTP persistente (*long-lived*) în vederea reducerii latenței



# interacțiune web: comet

Șablon de proiectare a aplicațiilor Web  
care necesită realizarea de conexiuni persistente,  
în stilul *peer-to-peer*

utilizat de aplicațiile Web intensiv interactive,  
eventual colaborative

exemple: Google Docs, Mibbit,...

# interacțiune web: comet

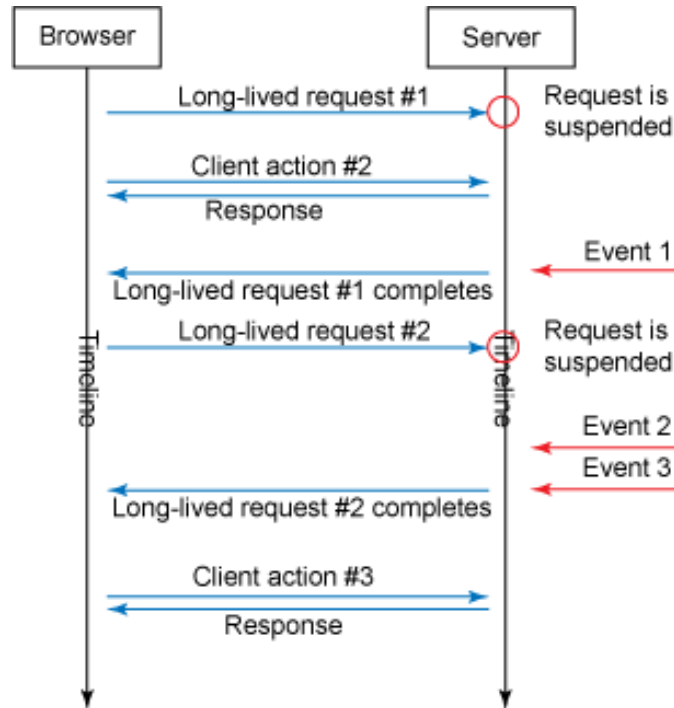
## Complementar Ajax

*long polling*

*HTTP server push*

*Reverse Ajax*

[http://ajaxpatterns.org/HTTP\\_Streaming](http://ajaxpatterns.org/HTTP_Streaming)



implementare: *HTTP long polling* sau *HTTP streaming*

de studiat M. Carbou, “*Reverse Ajax, Part 1: Introduction to Comet*”, IBM developerWorks, 2011  
<http://www.ibm.com/developerworks/web/library/wa-reverseajax1/>

# interacțiune web: comet

## Soluții de implementare

instrumente software – exemplificări:

Atmosphere, DWR, Ice Faces, Jetty, Orbited

în contextul JavaScript, un exemplu notabil este

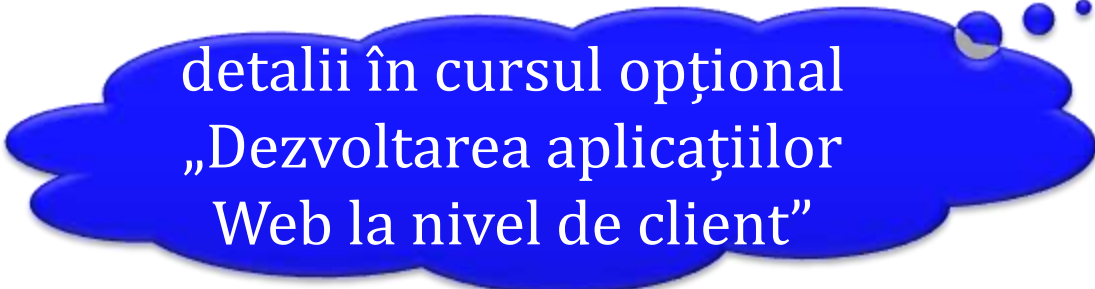
*APE (Ajax Push Engine)*

[www.ape-project.org](http://www.ape-project.org)

# interacțiune web: comet

Soluții alternative:  
adoptarea diverselor tehnologii HTML5

*server-sent events*  
*WebSocket*



detalii în cursul opțional  
„Dezvoltarea aplicațiilor  
Web la nivel de client”

# *mash-ups*

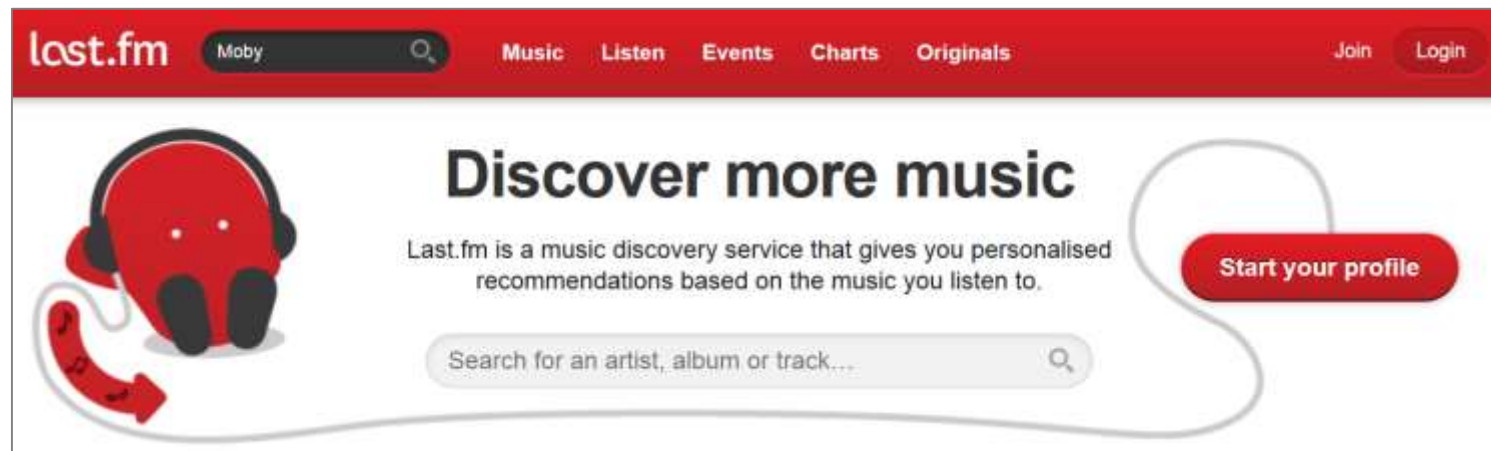
Ajax/Comet oferă suport pentru dezvoltarea de aplicații Web hibride – *mash-ups*

combinarea – la nivel de client și/sau server – a conținutului ce provine din surse (situri) multiple, oferind o funcționalitate/experiență nouă

# *mash-ups*

## Exemplificare:

dorim să oferim o aplicație ce pune la dispoziție  
informații din domeniul muzical  
în funcție de activitățile fizice ale utilizatorului,  
pe baza unor servicii Web publice



<http://www.last.fm/api/rest>



<https://wiki.fitbit.com/display/API/Fitbit+API>





## Account

Your API accounts

Add API account

## API Guides

Introduction

User Authentication

Scrobbling

Radio API

Feeds

Playlists API

Tools

REST requests

XML-RPC requests

Error codes

Terms of Service

## API Methods

### Album

[album.addTags](#)

[album.getBuylinks](#)

[album.getInfo](#)

[album.getShouts](#)

[album.getTags](#)

[album.getTopTags](#)

[album.removeTag](#)

[album.search](#)

[album.share](#)

### Artist

[artist.addTags](#)

[artist.getCorrection](#)

[Last.fm Web Services](#)

# REST Requests

The API root URL is located at <http://ws.audioscrobbler.com/2.0/>

Generally speaking, you will send a method parameter expressed as 'package.method' along with method specific arguments to the root URL. The following parameters are required for all calls:

**api\_key** : A Last.fm API Key.

**method** : An API method expressed as *package.method*, corresponding to a documented last.fm API method name.

For example:

```
http://ws.audioscrobbler.com/2.0/?method=artist.getSimilar&api_key=xxx...
```

If you are accessing a *write* service, you will need to submit your request as an HTTP POST request. All POST requests should be made to the root url:

```
http://ws.audioscrobbler.com/2.0/
```

With all parameters (including the 'method') sent in the POST body. In order to perform write requests you will need to authenticate a user with the API. See [authentication](#) for more.

## REST Responses

Responses will be wrapped in an lfm status node

```
<lfm status="$status">
  ...
</lfm>
```

acces la serviciile REST  
despre formatii + albume  
via o cheie de autentificare

Get user's *profile* in the *format* requested using *units* in the **unit system** which corresponds to the *Access Type*.

**Access Type:** Read

**Rate Limited:** Yes

**OAuth:** *oauth\_token* is optional, if omitted you should explicitly specify `<user-id>`.

**Privacy:** Basic profile is always public, **About Me** (Friends or Anyone), **Age and height** (Friends or other user's respective profile fields, considering:

- authenticated owner will receive all values, others will receive the correct values for accessible fields, empty string, "NA" (empty gender), 0 (empty height), default avatar etc., some values revealed

API-ul REST de la FitBit oferă  
date în formatele JSON și XML

## Resource URL

**GET** `/<api-version>/user/<user-id>/profile.<response-format>`

<b>api-version</b>	The API version. Currently 1.
<b>user-id</b>	User's encoded id or "-" (dash) to indicate user currently authenticated via the API.
<b>response-format</b>	The response format. Currently supported response formats are <b>json</b> and <b>xml</b> .

## Examples

GET `/1/user/228TQ4/profile.json`

GET `/1/user/228TQ4/profile.xml`

GET `/1/user/-/profile.json`

API

Fitbit

Service

https://api.

Select an API method

Search methods...

PROFILE

GET Get User Info

POST Update User Info

BODY

GET Get Body Measurements

GET Get Body Weight

GET Get Body Fat

GET Get Badges

GET Get Time Series

POST Log Body Measurements

POST Log Body Weight

POST Log Body Fat

DELETE Delete Body Weight Log

DELETE Delete Body Fat Log

# *mash-ups*

<http://www.last.fm/api/rest>



+



FiLa  
aplicație Web hibridă

<https://dev.fitbit.com/>

# *mash-ups*

Se bazează pe fluxuri de știri RSS/Atom,  
servicii Web, API-uri publice,...

„curentul” **SaaS** (*Software As A Service*)

# *mash-ups*

Caratteristici:

combinare

vizualizzare

agregare

# *mash-ups*

## Combinare

utilizarea de surse de date multiple  
poate avea caracter multidimensional

de exemplu,  
subiect de interes + locație geografică + moment de timp

*Yahoo! music search + Google maps + Eventful*



ubiGuide Places

Filtering nearby places...

Show me only cafes and museums...



## UbiGuide

An essential tool for any tourist, ubiGuide's unique capabilities give users an amazing Windows Phone 8 experience.

More Info at youtube [description](#).

Imagine Cup 2013 Project

Adviser by: PhD. [Sabin C. Buraga](#)

Students:

- Ionut Danila
- Mihaela Ghimiciu

proiectul **ubiGuide** (Ionuț Dănilă & Mihaela Ghimiciu, 2013—2014)

# *mash-ups*

## Vizualizare

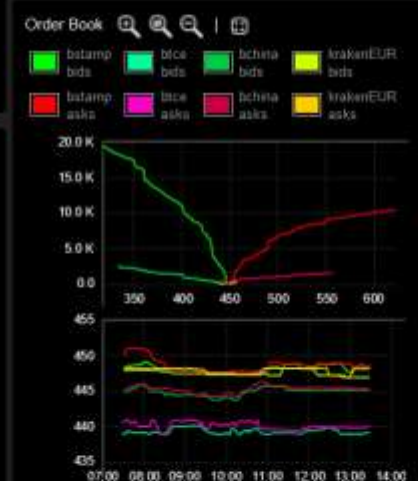
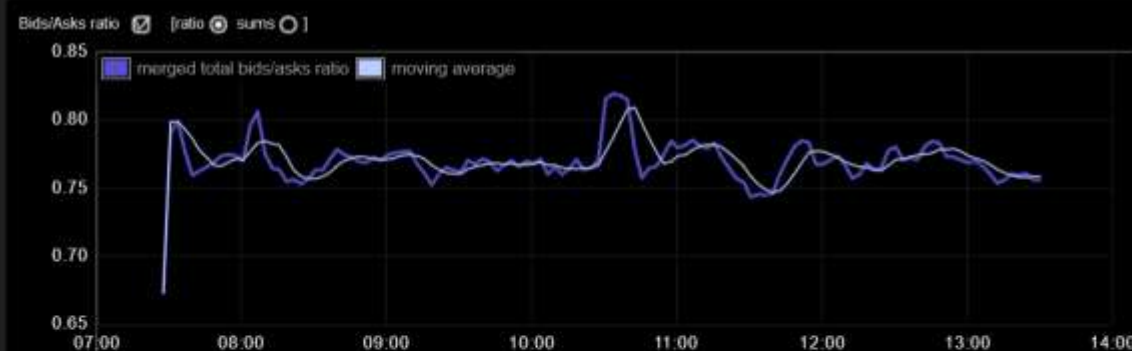
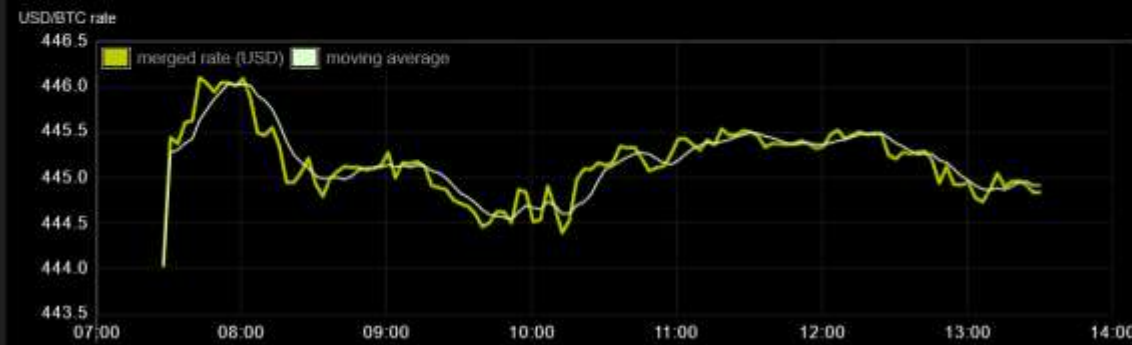
pot fi adoptate diverse tehnici de vizualizare  
(prezentare) a datelor:

*chart-uri*, cartografică, *tag cloud-uri*, tridimensională,...





MODE: single multiple **average** Bitstamp BTC-e Bitfinex Kraken ANX Huobi BTC-China ANX Kraken Bitcoin-Central  
 PERIOD: 10mn 1h **6h** 1d 1w 1m 6m 1y settings help



5/18/2014, 1:29:45 PM

- US Dollar \$ -  
 Bitstamp ↑ 448.33 \$ [18.7%]  
 BTC-e ↓ 439.01 \$ [8.1%]  
 Bitfinex ↑ 447.30 \$ [4.9%]  
 Kraken → 453.83 \$ [0.0%]  
 ANX → 450.09 \$ [0.1%]  
 - Chinese Yuan ¥ -  
 Huobi ↑ 2770.23 ¥ [53.0%]  
 BTC-China → 2775.00 ¥ [8.2%]  
 - HK Dollar HK\$ -  
 ANX → 3470.90 HK\$ [5.3%]  
 - Euro € -  
 Kraken → 326.30 € [1.4%]  
 BTC-Central → 330.00 € [0.4%]  
 Block: 301350  
 4 minutes 55 seconds ago  
 Difficulty: 8.85e+9  
 Target: 63.38 Phash/s  
 Current: 76.30 Phash/s  
 Next retarget in 1040 blocks  
 - 6 days

**Coinorama**: metode diverse de vizualizare în timp-real  
 a evoluției cursului monedelor virtuale

# *mash-ups*

## Agregare

gruparea datelor provenite din mai multe surse și  
analizarea lor: statistici, clasificări, predicții,...

*e.g.*, folosind *data mining* se pot releva  
aspecte „ascunse” ale datelor procesate

Enter title here

Permalink: <http://rekrealnica.wordpress.com/?p=162>


Add Media
Add Poll
Add Contact Form
Visual
Text

**B** *I* ABC


Format

**Link Love** is easy with Zemanta. We recommend related posts from other bloggers, you decide which ones you want to link to. We then let them know that you gave them some love. And of course, other bloggers can and will do that for you as well!


Related articles




Zemanta Raises Biggest



Upgrading the Zemanta




Book Bloggers New Years




Get Related Content for

Path: div.zemanta-pixie

In-Text Links

 Zemanta

 bloggers

APPLY ALL

Word count: 47





Draft saved at 1:45:24 popoldan.





Content Recommendations

Zemanta
My Sources
Sign In

Update

Media Gallery

Related Articles
with thumbnails

1 year ago
everything.typepad.com

My interview on Zemanta's blog
1 year ago
brandsandfilms.com

Book Bloggers New Years Challenge...Day 2
6 hours ago
thekindlequeen.wordpress.com

#BloggersLinkUp [NittyGrittyNesh & StyleForens
2 days ago
thestylingfirmblog.com

Get Related Content for Your Blog with the Zen
3 years ago
buzz.blogger.com

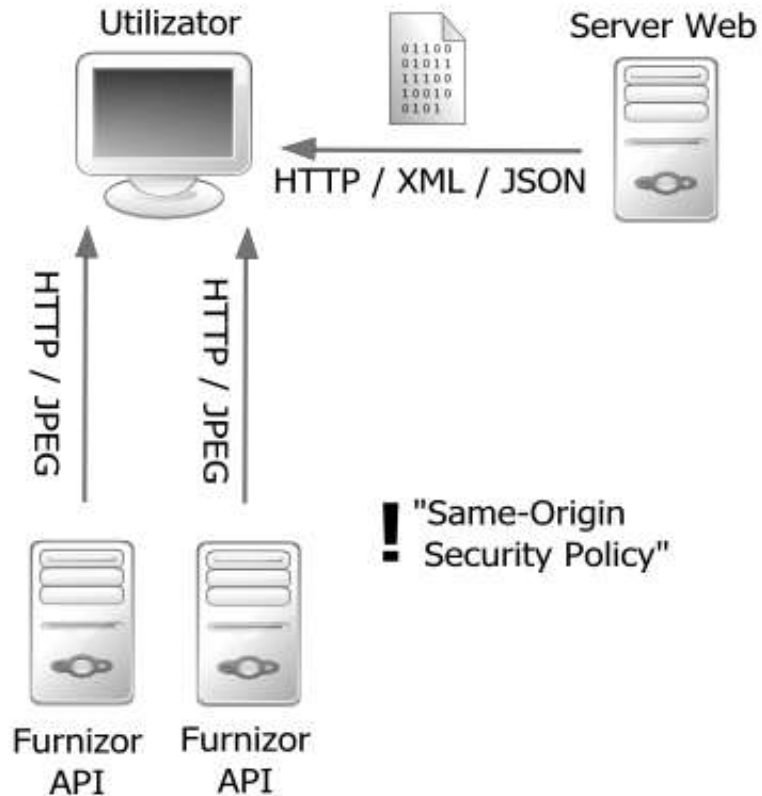
Columbus Blogger Meet Up
16 hours ago
teacherlingo.com

**Zemanta** – recomandare „inteligentă” de resurse

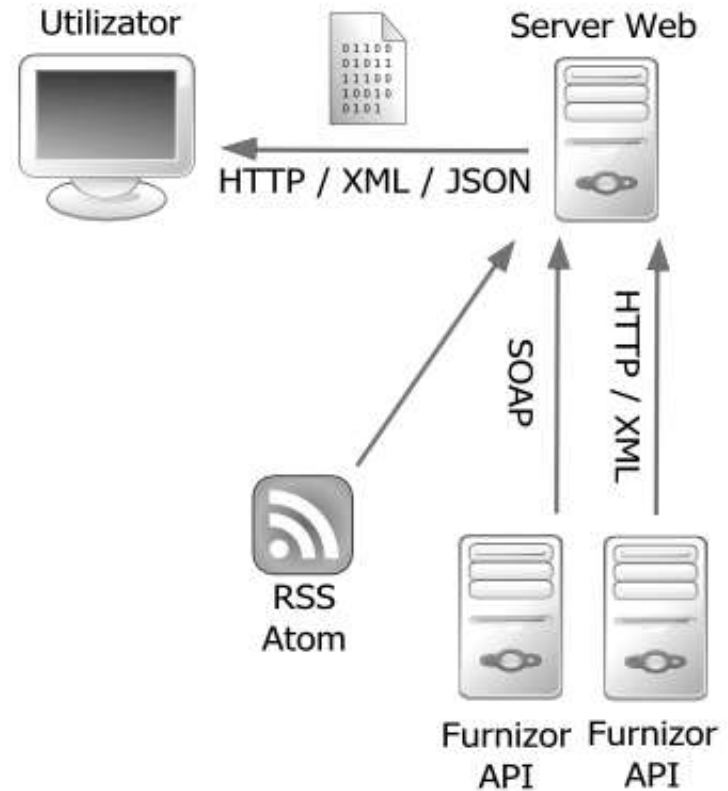
# *mash-ups*

Implementate la nivelul:  
clientului (*browser*-ului) Web  
și/sau  
serverului Web

## Abordare Client



## Abordare Server



# *mash-ups*

Surse de date ( <i>data feeds</i> )	Atom, RSS, geoRSS, microdate HTML5, RDFa,...
Interfețe de programare (API-uri)	specifice serviciilor publice și de procesare JSON/XML
Biblioteci/ <i>framework</i> -uri pentru dezvoltare	<i>framework</i> -uri Web generice sau oferite de organizații
Instrumente interactive ( <i>Web tools</i> )	eventual, disponibile în <i>cloud</i> <i>e.g.</i> , Yahoo! Pipes
Platforme ( <i>Platform As A Service</i> )	Heroku, Google App Engine, Nodejitsu, Windows Azure,...



# How we have changed the world

Refresh

## A TALE OF NEW CITIES

You are 2 years younger than  
Navi Mumbai

India



## ALL YOU CAN EAT

Amount available per person



2011

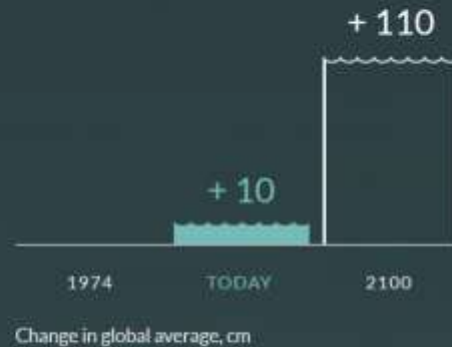
92

94

153

## SEA CHANGE

Rise of sea level



## POWER SUPPLY

Change in energy supply since 1980

COAL +114% RENEWABLES +133%



Marine fish caught

1950  
17.3

TODAY  
80.8

## LIFE ON THE EDGE

Number of plant species under threat



## EMISSION CONTROL

CO<sub>2</sub> emissions in your lifetime



*Your Life on Earth* (BBC, 2014)

[www.bbc.com/earth/story/20141016-your-life-on-earth](http://www.bbc.com/earth/story/20141016-your-life-on-earth)



# ProgrammableWeb: Showcasing Mashups and Apps by the APIs THEY CONSUME

Search Mashups

Filter Mashups

Search  
Security  
Semantic Web  
Semantics  
Sentiment  
SEO  
Shipping  
Smartphone  
SOA

By APIs

☐ Include Deprecated

	Category	Date
Chopper Is A Simple Demo Application	Tools	05.14.2015
Uses The WordPress.com API To Graph A Site's...		
Mapper By Nicholas Felton Is A	Fitness	04.28.2015
Sketch For Rendering Location And		
y Data...		
Serendipities App By Nicole Aptekar Is	Fitness	04.28.2015
Application That Allows Users To See Where		
they...		
Moves Modulate	Fitness	04.28.2015
Moves Elizabeth	Fitness	04.28.2015

API Directory Search

Search over 13,459 APIs  
updated daily



Browse by Category

Newest APIs

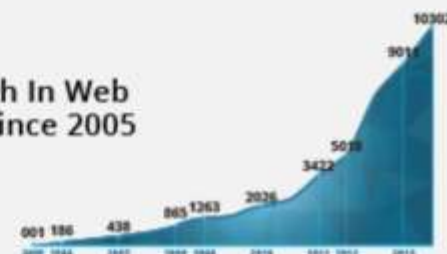
Latest Mashups

Add an API +

## PW Research Center

Our data. Your PowerPoints. Use our API research for your next presentation. [See all](#)

### Growth In Web APIs Since 2005



lista mash-up-urilor: [www.programmableweb.com/mashups/directory](http://www.programmableweb.com/mashups/directory)



# *mash-ups*: aspecte de interes

performanță: scalabilitatea și latența

limite ale API-urilor + existența versiunilor multiple

drepturi de autor asupra datelor & licențiere

securitate: abuz, confidențialitate, încredere etc.

monetizare

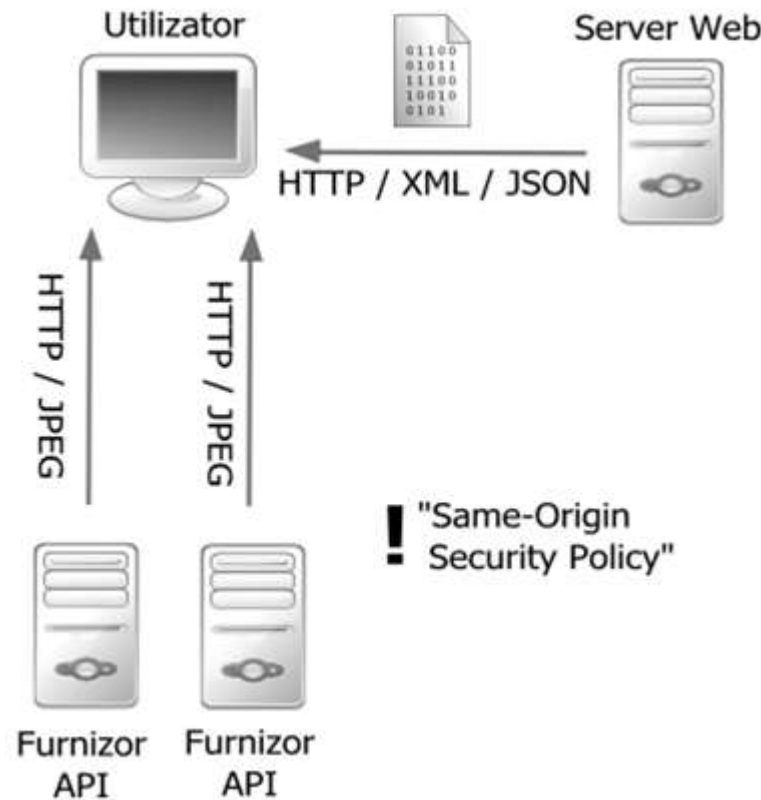
lipsa unei interoperabilități reale între platforme

Nu există o problemă de securitate  
privind accesul la resurse via JavaScript?

# *mash-ups: securitate*

## *Same-Origin Security Policy*

stipulează că un program JavaScript trebuie să acceseze doar datele aparținând aceleiași origini  
– *i.e.*, provenite din același domeniu Internet –  
a *script*-ului JavaScript



se permit doar transferuri vizând reprezentări de resurse referitoare la imagini, fișiere CSS și alte programe JavaScript aparținând aceleași origini

# *mash-ups: securitate*

## *Same-Origin Security Policy*

previne cazurile în care un document/program încărcat dintr-o origine să poată accesa/modifica proprietăți ale unui document aparținând altei origini

pentru detalii, a se consulta

[https://developer.mozilla.org/Web/Security/Same-origin\\_policy](https://developer.mozilla.org/Web/Security/Same-origin_policy)

```
var url = "http://profs.info.uaic.ro/~busaco/teach/courses/web/web-film";

// realizăm o cerere HEAD pentru a obține meta-date despre o resursă
var client = new XMLHttpRequest ();
client.open ("HEAD", url, true);
client.send ();
client.onreadystatechange = function () {
    // am recepționat câmpurile-antet?
    if (client.readyState == 2) {
        // semnalăm tipul MIME și data ultimei actualizări
        alert ("Resursa de tip " +
            client.getResponseHeader ("Content-Type") + " s-a actualizat la " +
            client.getResponseHeader ("Last-Modified"));
    }
}
```

preluarea în mod asincron  
via **HEAD** a unor meta-date

The image shows a web browser's developer console with the 'Console' tab selected. The console displays a red error message: 'Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at http://profs.info.uaic.ro/~busaco/teach/courses/web/web-film.html. This can be fixed by moving the resource to the same domain or enabling CORS.' The URL 'web-film.html' is highlighted in blue. Below this, a list of console messages is visible, including 'HTML1300: Navigation occurred.', 'SEC7118: XMLHttpRequest for http://profs.info.uaic.ro/~busaco/teach/courses/web/web-film.html required Cross Origin Resource Sharing (CORS).', 'SEC7120: Origin http://localhost not found in Access-Control-Allow-Origin header.', and 'SCRIPT7002: XMLHttpRequest: Network Error 0x80070005, Access is denied.' The console also shows the file 'ajax-head.html' associated with these messages. The target of the error is set to 'top: ajax-head.html'.

Console

Net CSS JS Security Logging Clear Filter output

⚠ Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at <http://profs.info.uaic.ro/~busaco/teach/courses/web/web-film.html>. This can be fixed by moving the resource to the same domain or enabling CORS. [web-film.html](#)

F12 Console [Error: 2] [Warning: 0] [Info: 2] [Log: 2] X

Target: top: ajax-head.html

- HTML1300: Navigation occurred.  
File: ajax-head.html
- SEC7118: XMLHttpRequest for http://profs.info.uaic.ro/~busaco/teach/courses/web/web-film.html required Cross Origin Resource Sharing (CORS).  
File: ajax-head.html
- SEC7120: Origin http://localhost not found in Access-Control-Allow-Origin header.  
File: ajax-head.html
- SCRIPT7002: XMLHttpRequest: Network Error 0x80070005, Access is denied.  
File: ajax-head.html

URL al altui domeniu ► eludăm *Same Origin Policy*

# *mash-ups: securitate*

## **CORS** (*Cross-Origin Resource Sharing*)

recomandare W3C – ianuarie 2014

<http://www.w3.org/TR/cors/>

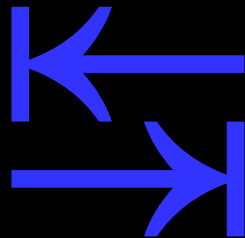
mecanism ce permite partajarea la nivel de client  
a resurselor provenind din domenii Internet diferite

astfel, se pot emite cereri via XMLHttpRequest între domenii

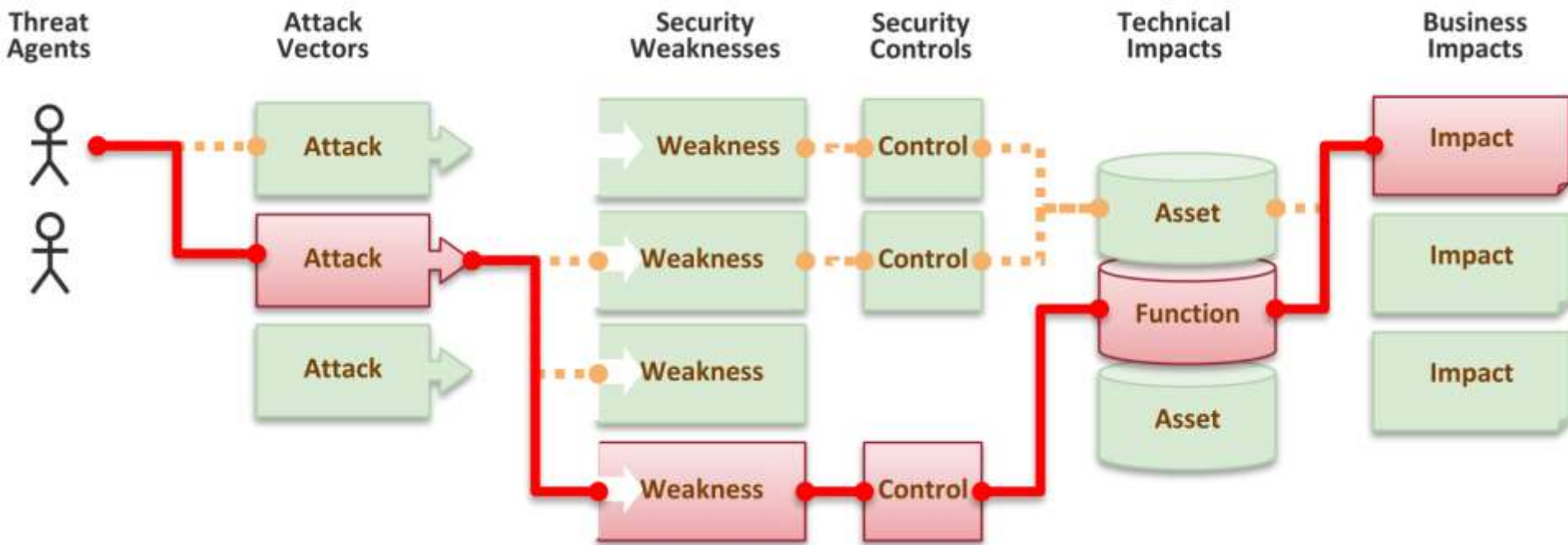


# rezumat

## interacțiune Web



transferul asincron al datelor  
de la Ajax la *mash-up*-uri Web



„ultimul” episod: **securitatea aplicațiilor Web**