AJAX Framework építés

Nagy Attila Gábor
Wildom Kft.
nagya@wildom.com

Mi az AJAX?

- Asynchronous JavaScript and XML
- Ennél azért kicsit több:
  - Koncepció váltás a felhasználói interface tervezésben
  - Standard kompatibilis HTML!
  - XML-en kívül más adatformátumok is (sőt!):
    - JSON (JavaScript Object Notation)
    - Plain text
    - HTML

AJAX vagy hagyományos HTTP?

AJAX

- Kisebb sávszélesség és CPU terhelés
- Alkalmazás logikai és prezentációs réteg jobb elválasztása
- Kliens oldali logika: tárolt adatok, navigáció követhető
- Kényelmesebb felhasználói felület

HTTP

- Bevált eszközök: IDE, framework, template engine
- Böngésző független
- Ismert erőforrások
- Kereső motorok könnyebben tudják indexelni

Miért nem használtuk eddig?

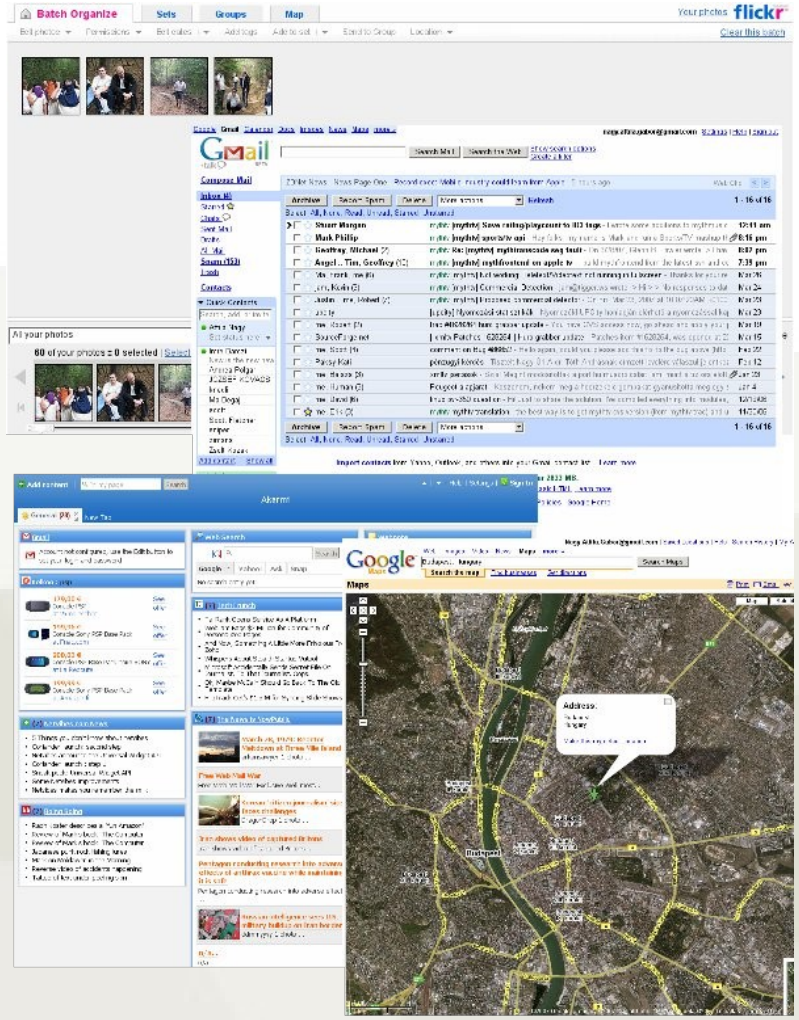
- Böngésző inkompatibilitás
- Kezdetleges JavaScript (fontos eszközök hiánya)
- Komoly JavaScript futtatásához lassú gépek és böngészők
- Nem megfelelően alkalmazott HTML
- Lassú internet kapcsolatok (rossz válaszidők)

Mi változott?

- Erősebb gépek
- Standard kompatibilis HTML, XHTML
- Letisztult JavaScript, jól használható DOM specifikáció
- Ezekre épített cross-browser alkalmazás könyvtárak
- Gyorsabb internet (jobb válaszidők)

Vonzó példák

- Meglepően jól működő rendszerek:
  - gmail
  - netvibes
  - flickr
  - del.icio.us
  - stb
- Én is akarok ilyet!



Mire használnak AJAX-ot?

Kezelhetőség segítése

- flickr adatlapok
- del.icio.us
- iwiw

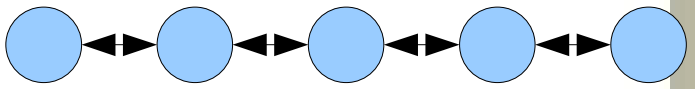
Desktop imitálása

- gmail
- google maps
- flickr képrendező
- netvibes

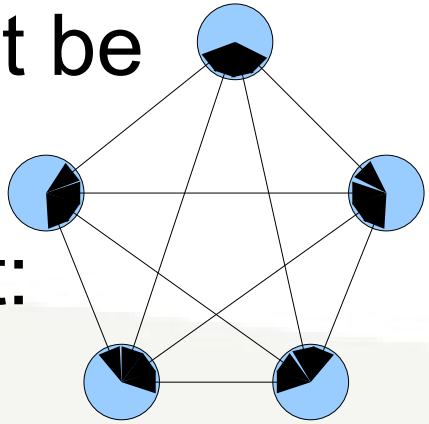
Konkrét esetben

- Volánbusz on-line jegyrendelő rendszer
- Sok bonyolító tényező:
  - nehezen megkülönböztethető termékek (jegyek)
  - szigorú kontingens kezelés - zárolások
  - több ajánlat készítése
  - on-line fizetés (nem csak rendelés)
- Step-by-step, varázsló jellegű működés

Miért AJAX?



helyett



- A hagyományos működés nem vált be
- A szabad előre-hátra lépkedés nagyon megbonyolítja a rendszert:
  - zárolások kezelése nehézkes
  - az oldalak nem támaszkodhatnak egymásra
- A rendszert több helyre is be kellett illeszteni, ezért függetleníteni kellett a weboldal rendszerétől
- A varázslók tipikus asztali alkalmazások

Mit használjak?

Teljes körű könyvtárak

- Yahoo UI
- Google Web Toolkit

Túl összetett (megtanulni)

Alapkönyvtárak

- Prototype
- jQuery
- advAJAX
- TrimPath

Egyik sem tud mindent

Rakjuk össze darabonként

- Az alapkönyvtárak jól kiegészítik egymást
- Együtt használva sem túl nagyok
- A kiválasztott eszközök:
  - Prototype
  - advAJAX
  - TrimPath JavaScript Templates

Mire kell: Prototype

- Legalsó réteg
- Böngésző függetlenség kialakítása
- Eseménykezelés megoldása
- Egyszerűsíti a programozást, olvashatóságot

Mire kell: advAJAX

- JavaScript és a webszerver közötti kommunikációra
- Használhattuk volna a Prototype szolgáltatásait is, de ez szerintem:
  - Egyszerűbben használható
  - Több szolgáltatást nyújt:
    - Összetett hibakezelés
    - Egy időben futó kérések összehangolása
    - Jó paraméterezhetőség

Mire kell: JavaScript Templates

- A szerver válasza alapján összetett oldalakat kell előállítani:
  - Táblázatok, listák
  - Űrlapok
- Tisztán a DOM csak apró módosítások esetén kényelmes
- Szintaktikája hasonló a Smartyhoz
- Megjelenítés elválasztható a logikától

JST lehetőségei

```
<h1>Személy:</h1>
<p>Név: ${user.name}</p>
<p>Cím: ${user.address}</p>
<p>Házas: {if user.married}igen{else}nem{/if}</p>
<h1>Gyermekek:</h1>
<ul>
  {for child in user.children}
  <li>${child.name}</li>
  {forelse}
  <li><em>nincs</em></li>
  {/for}
</ul>
```

Oldalak leírása

- A keretrendszer bemenete
- Leegyszerűsíti egy oldal betöltését

```
var PAGES = { 'index':  {template: 'index.tpl'},
              'search': {template: 'search.tpl'},
              'data':   {template: 'data.tpl'}
            };
```

Oldal betöltés

- A fenti oldalleírás alapján letöltjük az oldalt

```
function loadPage(name, contentData) {
    var pageData = PAGES[name];
    advAJAX.get({url: TEMPLATE_ROOT+pageData.template,
                 onSuccess: initialiseTemplate,
                 onError: errorHandler,
                 tag: {'pd': pageData}});
}
```

Oldal megjelenítés

- TrimPath Template objektum előállítása

```
function initialiseTemplate(obj) {
    var template = TrimPath.parseTemplate(obj.responseText);
    intialisePage(template, obj.tag.pd);
}
function initialisePage(template, pd) {
    $(TARGET_NODE_ID).innerHTML =
          template.process(pd.contentData);
}
```

Testreszabhatóság 1.

- Hookok bevezetése

```
var PAGES = { 'index':  {template: 'index.tpl'
                         init: function() {...},
                         onload: function() {...}},
              'search': {template: 'search.tpl'},
              'data':   {template: 'data.tpl'}
            };
```

Testreszabhatóság 2.

- Hookok bevezetése

```
function initialisePage(template, pd) {
    // init hook hívása
    var parameters =
      pd.init ? pd.init(pd.contentData) : pd.contentData;
    $(TARGET_NODE_ID).innerHTML =
          template.process(parameters);
    // onload hook hívása
    if (pd.onload) pd.onload();
}
```

További egyszerű bővítések

- Betöltött template-ek cachelése memóriába, így a kliens és a szerver terhelése is csökkenthető
- Globális paraméterek: az initialisePage minden oldalletöltésnél hozzáadja a paraméterekhez (például képek elérési útja, nyelv beállítás, stb)
- onunload kezelése, például zárolások feloldására.
  - Nagyon megbízható!
  - pl: `<body onunload="releaseLocks()">`

Események kiváltása

- Az oldalon található vezérlők indítják el a funkciókat
- <a> tag nem jó: kivezet az oldalról
  - ha a metódus nem tér vissza false értékkel
  - ha a kódban hiba történik
- Helyette tagek használata célszerű:

```
<span id="next"
      class="link"
      onclick="loadPage('kovetkezo');">tovább</span>
```

Vezérlők automatikus elhelyezése

- Így a template-be egyáltalán nem kell JavaScriptet írni
- Oldalleírás:

```
var PAGES = { 'index':  {template: 'index.tpl'
                         init: function() {...},
                         onload: function() {...},
                         hotspots: {'prev': function() {...},
                                    'next': function() {
                                         loadPage('nextpage');
                                    }},
            };
```

Vezérlők automatikus elhelyezése

- onclick események definiálása az oldal betöltése után:

```
function initialisePage(template, pd) {
    (...)
    $(TARGET_NODE_ID).innerHTML =
         template.process(parameters);
    for (var i in pd.hotspots) {
        var hotspot = $(i);
        if (hotspot} hotspot.onclik = pd.hotspots[i];
    }
    (...)
}
```

Előre, hátra gombok kezelése

- Felhasználó nagyon hozzászokott
  - ha baj van, azonnal a vissza gombot nyomja meg
- Bonyolult feladat:
  - Eseményként nem kapható el
  - Böngészőnként változó megoldás
    - Firefox, Safari: url hash részének figyelése egy időzítővel. Könnyű, megbízható, és elegáns.
    - IE: rejtett iframe-be töltünk egy oldalt, ami visszahívja a szülő alkalmazást. Nehézkes, és bizonytalan.
  - Bővebben: http://www.contentwithstyle.co.uk/Articles/38/

Architektúrális felépítés

Szerver

- Adatbázis
- Alkalmazás logika
- API jellegű interface a JavaScript felé
- Session kezelés
- Template-ek előkészítése (például i18n)

Kliens

- Állapotgép, eldönti mely API hívások szükségesek
- Megjelenítés
- Felhasználói input előszűrése, és eljuttatása a szerver felé

Adatkommunikáció

- Tartalmak: HTML, plain text
- Adatok: JSON
  - XML: nehezebb kezelni szerver és kliens oldalon is
  - JavaScript oldalon egy eval
  - PHP implementáció: http://mike.teczno.com/json.html
- Érdemes egy réteget készíteni erre
  - Adatok hozzáadása, küldése
  - Hiba küldés

Biztonság

- Kliensben sosem bízhatunk meg!
- Többet tudó kliens, ezért jobban kell figyelni
- Klienstől jövő adatokat akkor is ellenőrizni kell, ha azt a JavaScript már megtette
  - Adatok szintaktikai helyessége
  - Helyes API hívási sorrend
  - Apró hiba esetén is mindent megállítani!
  - Egyszerűsödő szerver: használjuk ki ellenőrzésre!

Biztonság

- Csak azt adjuk ki a JavaScript felé, amire okvetlenül szüksége van
- API metódusok jól körülhatárolható funkciókat lássanak el, például:
  - Visszaadja a célállomások nevét
  - Útvonalat keres két település között
  - Adott napra kikeresi a járatokat
  - A sessionben tárolt járatokra megadja a jegyárat

Linkek

- Prototype: http://www.prototypejs.org/
- jQuery: http://jquery.com/
- advAJAX: http://advajax.anakin.us/index-en.htm
- TrimPath JST: http://trimpath.com/project/wiki/JavaScriptTemplates
- Yahoo! UI: http://developer.yahoo.com/yui/
- Google WebToolkit: http://code.google.com/webtoolkit/