Web2.0-ás alkalmazások mobil környezetben

NAGY GÁBOR, SCHULCZ RÓBERT

Budapesti Műszaki és Gazdaságtudományi Egyetem, Híradástechnikai Tanszék
schulcz@hit.bme.hu

Kulcsszavak: Web2.0, Ajax, mobiltelefon-böngészők

Manapság egyre fontosabb, hogy a megszokott alkalmazásainkat akár mobiltelefonon is elérhessük, melybe természetesen beletartozik az összes általunk kedvelt weboldalt meglátogatása is. A Web2.0-ás alkalmazásoknál már a felhasználó is részt vesz az oldalak tartalmának szerkesztésében. Technikai oldalról ennek egyszerű megvalósítását szolgálja az Ajax fejlesztési technika is. Éppen ezért döntöttünk úgy, hogy megvizsgáljuk ennek működőképességét mobil környezetben, melynek legegyszerűbb módja a telefonok böngésző alkalmazásainak tesztelése.

1. Bevezetés

Természetesen minden készülék tesztelésére nincsen lehetőség, de igyekeztünk úgy választani, hogy a jelenleg használatos készülékek közül minél többet kipróbáljunk. Mivel egy-egy újabb készülék az előzőtől böngésző szempontjából nem különbözik, így négy-öt készülékkel a teljes spektrum nagyobb része lefedhető.

Az okostelefonok kezdetben a szűk piaci részesedésük miatt nem fektettek nagy hangsúlyt jó minőségű kliensalkalmazás fejlesztésébe, vagy már egy létező böngésző mobil változatának elkészítésébe. Így fordulhatott elő, hogy tényleg csak a megjelenítéshez szükséges alapfunkciókat implementálták az adott szoftverben és az esetlegesen szükséges további webes technikák megvalósítása, illetve kidolgozása már nem kapott akkora szerepet.

2. Futási feltételek

Az Ajax (Asynchronous JavaScript and XML) interaktív webalkalmazások létrehozására szolgáló webfejlesztési technika. A weblap kis mennyiségű adatot cserél a szerverrel a háttérben, így a lapot nem kell újratölteni minden egyes alkalommal, amikor a felhasználó módosít valamit. Ez növeli a honlap interaktivitását, sebességét és használhatóságát.

Az Ajax technológia a következő szabványokat használja fel:

- XHTML (vagy HTML) és CSS:
  a tartalom leírására és formázására szolgál.
- DOM (Document Object Model) szabvány:
  célja, hogy a kliens oldali programozási nyelvekkel egyszerűbben/egyértelműbben hivatkozhassunk a dokumentum elemeire a dinamikus oldalaknál.
- XMLHttpRequest objektum:
  az adatok aszinkron kezelésére szolgál a kliens és a webszerver között.
- XML szabvány:
  a kliens és szerver közötti adatok továbbítására szolgál, ám helyettesíthető sima szöveggel, vagy HTML adatokkal is.

Tehát a készüléknek ezeket ismernie kell a megfelelő működéshez.

3. A tesztelés

Az alábbiakban összefoglaljuk a futáshoz szükséges feltételek ellenőrzésének lépést. A lépesek egymásra épülnek, a legalsó szintről haladunk fölfelé, feltérképezve ezzel a futtatott szoftver képességeit.

3.1. XHTML kompatibilitás teszt

Először is le kell ellenőriznünk, hogy az általunk használni kívánt készülék képes-e megjeleníteni az XML alapú HTML oldalakat. Ezzel megbizonyosodhatunk afelől, hogy az általunk elképzelt oldalképet meg tudja majd jeleníteni, valamint, hogy a technológia nevében szereplő XML kompatibilitásnak eleget tesz.

A W3C [1] által kiadott szabványnak megfelelő egyszerű weboldalt hozunk ehhez létre, amely az alapelemeket tartalmazza példaképpen, valamint a fájl szintaktikailag megfelel a kiadott dokumentációnak, tehát valid.

Az oldal megnyitásával a böngészőben könnyen meggyőződhetünk a készülék alkalmasságáról.

3.2. JavaScript kompatibilitás teszt

Az előzőleg létrehozott dokumentum fejrészében hozzáadjuk a JavaScript függvényt tartalmazó részt, amiben egy egyszerű funkciót valósítunk meg, például egy felugró ablakot hozunk létre. Ezt a függvényt fogjuk meghívni a dokumentum törzsében (body részében) a betöltődést figyelő eseménykezelővel. Ennek az eredménye az lesz, hogy ha megnyitjuk ezt a weboldalt és egy felugró ablak üdvözöl bennünket, akkor a böngésző JavaScript kompatibilis.

A JavaScript fontossága az Ajax-technológia szempontjából elengedhetetlen, mivel segítségével tudjuk figyelni a felhasználói eseményeket, és tudunk ezekre a háttérben reagálni anélkül, hogy a weboldalt újra letöltetnénk a klienssel a kiszolgálóról. Ilyen esemény lehet például egy mező tartalmának lecserélése, új kép betöltése, keresési mező tartalmának elküldése.

3.3. DOM ismeret tesztelése

A JavaScript több, mint egy évtizedes múltra tekint vissza, de áttörése csak az utóbbi pár évben történt meg. Ennek oka, hogy a kifejlesztésének idején nem történt meg a szabványosítás, így a különböző böngészők (sokszor szándékos) inkompatibilitásokat vittek bele a nyelv implementálásába, ezzel megakadályozva az egyes elemekre való egyforma hivatkozást a különböző szoftverek alatt.

A W3C által kiadott DOM (Document Object Model) éppen emiatt egységesíti, hogyan is kell JavaScript-ben az egyes elemekre hivatkozni. A felépítés tartalmazás szerint halad befelé. A legkülső elem a dokumentum maga mindig. Ez az Ajax szempontjából különösen fontos, hiszen a használata során az oldal bizonyos elemeit változtatjuk csak meg, tipikusan frissítjük azokat. Ehhez elengedhetetlen, hogy ezt minden böngésző alatt egyformán tudjuk megtenni és ne kelljen azzal külön foglalkoznunk, hogy az eltéréseket kezeljük, mert az erőforrás-pazarlást jelentene, különösen a mobil környezetben.

A tesztelés során egy szöveges beviteli mező tartalmát próbáljuk meg megjeleníteni egy felugró ablakban. Ezzel vizsgálhatjuk azt, hogy ki tudjuk-e olvasni az elemek tartalmát, tehát tudunk-e azokra hivatkozni.

3.4. XML csomópontok kezelése

Az előbbiekben bemutatott szabványnak köszönhetően nem csak hivatkozni tudunk JavaScript segítségével elemekre, hanem azokat működés közben létre is tudjuk hozni és a dokumentum tetszőleges csomópontja alá, mint gyerekcsomópontot oda tudjuk csatolni. Ennek köszönhetően könnyen létrehozhatunk olyan táblázatot, amit a felhasználó kényelmesen tud bővíteni anélkül, hogy újra kellene tölteni a nézett honlapot.

Ezen funkció kipróbálására létrehoztunk egy olyan dokumentumot, amelyben az elhelyezett táblázat tartalmaz egy sort, majd a mellette lévő gomb segítségével újabb számozott sorokat adhatunk hozzá.

3.5. A nem szabványos InnerHTML kezelése

A szabványban a legtöbb elemnek nincs definiálva ilyen tulajdonsága, ám a technológiát alkalmazó weboldalak legtöbbje előszeretettel használja azt. Ennek oka, hogy segítségével egyszerűen lehet kicserélni a HTML elemnek a teljes tartalmát egy karakterláncra, amely akár új elemeket is tartalmazhat. Használatát az indokolja, hogy a legtöbb elterjedt desktop-böngésző támogatja, valamint a működése gyorsabb, mint az előbb bemutatott módszernek, továbbá ilyen alkalmazási módnál nem kell előzőleg feldolgozni a kapott adatokat, hanem csak értékül kell adni az elemnek, tehát sokkal rövidebb kódot igényel.

A tesztelés során egy szöveges beviteli mező tartalmát próbáljuk beírni a mellette lévő gomb megnyomásának hatására az alatta található rétegbe. Jelen teszt csak a rétegek működését vizsgálja meg, ugyanis ez a leggyakoribb elem, amit frissíteni szoktak az Ajax használata során.

3.6. A nem szabványos XMLHttpRequest ismerete és kezelése

Ezek után meg kell azt vizsgálnunk, hogy a technológia lelkét képező, ám jelenleg még nem szabványosított objektum létrehozására alkalmas-e a telefonon futtatott böngésző. Ehhez elegendő az objektumból egy példányt létrehoznunk a használathoz. A tesztkörnyezet ennek sikerességét vizsgálja, ha nem létezik ilyen objektum, akkor kivételt generál, amit jelez a felhasználó felé.

Enélkül az objektum nélkül nem tudunk a kiszolgálóról az alkalmazás segítségével a háttérben aszinkron módon adatokat letölteni.

3.7. Aszinkron GET és POST művelet elvégzése

Ha már birtokában vagyunk a szükséges feltételeknek, akkor ideje kipróbálni azt. A GET típusú kérések fordulnak elő a leggyakrabban, mikor az oldal egy részéhez új adatokat kérünk le a kiszolgálóról. A POST művelet során a küldött paramétereket nem a lekérdezésbe magába rakjuk bele, hanem mellette egy külön karakterláncként. Ezt leggyakrabban űrlapok elküldésére használják, tipikusan a bejelentkezéseknél.

Ajax

Az Ajax jelentése „Asynchronous JavaScript and XML”, azaz „Aszinkron JavaScript és XML”. A kifejezés először egy 2005. februárjában megjelent cikkben fordul elő, de a technika alapjai jóval régebbre nyúlnak vissza.

A hagyományos böngésző-webszerver modellben a böngésző mindig kérések formájában kéri le a tartalmat, és az egész oldat újratölti. Az első próbálkozásokat a böngészők egy-egy új elem bevezetésével tették meg, így az oldal egy része frissíthetővé vált anélkül, hogy a teljes oldalt újra le kellett volna tölteni (1996-ban az Internet Explorer 3 és 1997-ben a Netscape 4). Mindkét elemtípus esetében lehetséges ezek JavaScriptből történő módosítása, így már elhárult az akadály az oldal egyes részeinek frissítése elől. Az igazi áttörést azonban az jelenti, amikor JavaScript segítségével a háttérben kéréseket is tudunk küldeni a szervernek, amire válaszként valamilyen (legtöbbször XML) üzenetet kapunk, melyet JavaScript segítségével feldolgozunk. Az első megvalósításokban egy Java applet kérte le az adatokat, amivel a kliens oldal JavaScripten keresztül tudott kommunikálni, majd a 2002-ben létrejött „XMLHttpRequest” segítségével már közvetlenül lekérdezhetővé váltak az adatok. Ez a fontosabb böngészőkben hamar elérhetővé vált, így ma már ezt használják mindenhol.

3.8. A Framework-ökkel való kompatibilitás

A technológia terjedésével együtt megjelentek a különböző fejlesztést segítő keretrendszerek. Ezek hivatottak kezelni egyes nem szabványos elemekből adódó különbségek áthidalását a böngészők között, ezzel könnyítve a fejlesztő dolgát, hogy ténylegesen csak a kitűzött feladatra koncentrálhasson, valamint különböző előre elkészített funkciókkal segítik a gyorsabb fejlesztés menetét.

Az egyik legelterjedtebb és legletisztultabb ilyen keretrendszert tettük próbára; a Prototype JavaScript Framework-öt. Létrehoztunk egy olyan honlapot is, ahol a korábban bemutatott frissítést ezen keretrendszer segítségével tesszük meg.

4. Tesztelt eszközök

A tesztelés során igyekeztünk minél több olyan eszközt választani, ahol nem egy elterjedt böngésző mobil változatát mellékelik a felhasználók számára, hanem valamilyen a cég által fejlesztett egyedi szoftvert. Ennek oka az, hogy azon alkalmazások mobil változatai is nagyobb valószínűséggel vannak felkészítve a technológia helyes futtatására, mint a korábban tényleg csak mobil környezetre kialakított programok.

4.1. Nokia N91 és N95

A két telefonon ugyanaz a Nokia által készített böngésző alkalmazás található meg, csak az azt futtató Symbian operációs rendszer verziójában, illetve az azt kiszolgáló hardverben van különbség.

A teszt lépéseit végigpróbálva mindkét telefonon az összes funkciót működőképesnek találtuk. A képernyőn a honlapból csak az ablakméretnek megfelelő részt láthatunk, de a mutató segítségével ezt minden irányban mozgathatjuk, így megtekinthetjük az egész oldalt.

4.2. Sony Ericcson V640i

A készüléken SE live! platformon fut a Netfront által készített böngésző. Az alkalmazás honlapján láthatjuk, hogy külön kiemelik az Ajax támogatását. A gyakorlatban is minden ponton tökéletesen teljesített az eszköz. A böngésző két módban is használható, egyikben a CSS fájlokat felülbírálva a képernyő szélességére igazítja honlap megjelenítését, a másikban pedig a készítő által tervezett látványképet kapjuk. A szoftver támogatja az összes webes szabványt, így a böngészés során nem kell korlátokba ütköznünk.

4.3. Nokia 7650

A készülék az első teszten még túljutott és megjelenítette a tesztoldalt, ám JavaScript futtatására már nem alkalmas, így sajnálatos módon az ajaxos oldalakkal sem bírkózik meg.

4.4. Motorola V3xx

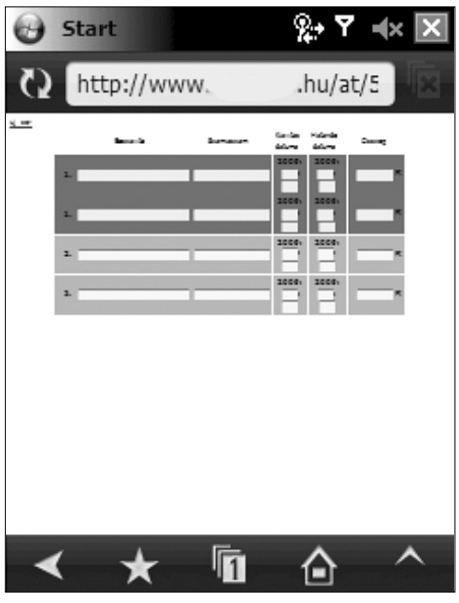
A telefon specifikációja szerint Opera 8.0 böngészőt futtat. A tesztpontok, az utolsó kivételével (framework kompatibilitási teszt) sikeresen lefutottak. Láthatóan alkalmas lehet az ajaxos oldalak megtekintésére. Érdekessége, hogy a JavaScript kódban szereplő XHTML kódot is ellenőrzi futás közben és hajlamos arra, hogy emiatt XML értelmezési hibával elutasítsa a megjelenítést.

Egy másik tapasztalat, hogy éppen ezért nem veszi figyelembe, ha valami megjegyzésként szerepel a Java Script forráskódban, így akkor is jelentkezhetnek XHTML értelmezési hibák. Ez inkább fenyegetést jelenthet a használat során, mivel sajnálatos módon kevés honlap ügyel arra, hogy a W3C által kiadott szabványnak megfelelő oldalt alakítson ki, mivel ezt a böngészők közötti differencia is nehezíti, továbbá a legapróbb gépelési hibák miatt is értelmezhetetlenné válhatnak az oldalak, amikre egyébként nem derül fény a tesztelés során.

4.5. Opera Mini [2]

Az Opera böngésző Opera Mini változata szabadon letölthető a hivatalos honlapjáról az összes olyan mobiltelefonra, amelyek Java futtatására alkalmasak. A telepítés után a Java-s alkalmazások között található meg. Képes használni a virtuális gépen keresztül a telefon által biztosított hálózati kapcsolatokat. Mivel a szoftver kifejezetten ezt a környezetet célozza meg, így minden szükséges funkciót implementáltak benne, és erről a tesztelés során is meggyőződtünk, minden lépés sikeresen végrehajtódott *(1. ábra)*.

1. ábra
XML csomópontok kezelése
Opera Mini alatt

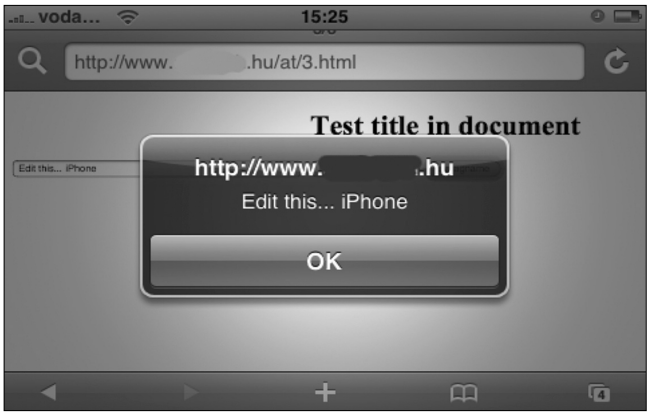


4.6. MiniMo [3]

A Mozilla projekt kifejezetten mobil eszközökre szánt böngészője. A fejlesztés 2004-ben indult meg és még 2005-ben megjelent a 0.1-es változat. Akkor még a Nokia támogatta, ám egy évvel később saját böngésző fejlesztéséről számolt be, amihez a KHTML motort választotta. Ezek után a szoftverfejlesztés szinte teljesen le is állt, nem készült belőle hivatalos bináris kiadás, aki ki szeretné próbálni, annak saját magának kell a csomagot előállítania. A hivatalos weboldal szerint az egyetlen támogatott platform a Nokia 800-as, illetve 810-es telefonjai. Készülék hiányában, valamint a projekt „kezdeti” stádiuma miatt nem teszteltük.

4.7. Apple iPhone és Safari

Az Apple által kiadott készülék egy teljesértékű Mac OSX operációs rendszert futtat a gyártó állítása szerint, így a vele érkező böngé-



2. ábra Input mező tartalmának megjelenítése Safari-ban

sző alkalmazás is az. A tesztek alapján minden lépést sikeresen végre is tudtunk hajtani vele, ugyanúgy, ahogyan azt a Macintosh-okon és Microsoft Windows rendszereken futó változataival is *(2. ábra)*.

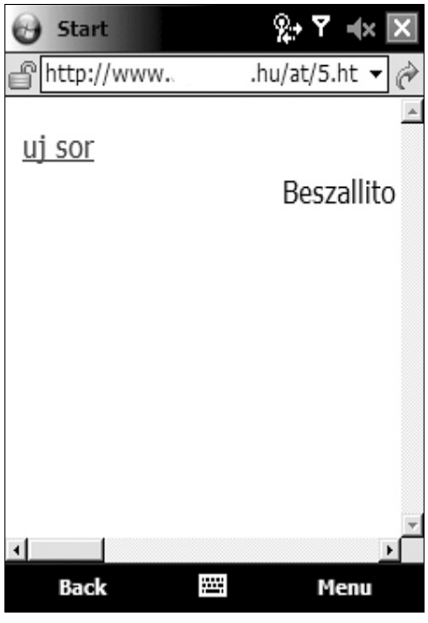
4.8. Windows Mobile és Internet Explorer

A legtöbb PDA telefonnal együtt érkezik a Microsoft Windows Mobile valamely verziója és a hozzá tartozó Internet Explorer. A jelenleg elérhető legfrissebb verzió a 6.1, viszont ez kevesebb készüléken található meg jelenleg, mint 6.0-ás elődje. A két változat csak nagyon minimális különbséggel bír, ám ez érinti az Internet Explorer-t is, így leteszteltük mindkét verziót, valamint összehasonlításképp megvizsgáltuk az 5.0 változatot is.

Ez volt az első olyan alkalmazás, ami nem teljesítette tökéletesen az általunk megszabott tesztfeltételeket. Először a DOM tesztnél akadt meg, ahol a csomópont neve alapján nem tudta megtalálni azt, az 5-ös verzió pedig se csomópont típusnév alapján, sem név alapján nem tudott hivatkozni az elemre. A következő hiányossága az volt, hogy nem tudott XML csomópontot létrehozni és hozzáfűzni az adott dokumentumhoz *(3. ábra)*, valamint a legutolsó tesztlépésben sem sikerült a Prototype JavaScript Framework segítségével betöltenie a kért oldalt.

Ettől függetlenül sok esetben még akár működhetnek is vele az ajaxos weboldalak, ugyanis a legáltalánosabb felhasználás – amikor egy saját script segítségével a fejlesztő frissíti egy réteg tartalmát –, a böngészőben sikeresen megvalósult.

3. ábra
XML csomópontok kezelésének hiánya Internet Explorer-ben



5. A mindennapi használat

5.1. iWiW [4]

Utolsó lépéseként a legnépszerűbb magyar közösségi oldal működését is megvizsgáltuk minden készülék alatt. A honlap üzemeltetői már felkészültek mobil kliensek fogadására is, létrehozva számukra egy ilyen változatot. Amikor a weboldalt megnyitottuk a mobil böngésző alkalmazásokban, automatikusan átirányítottak a mobil készülékekre szánt változatra [5]. Láthatóan a készítők felkészültek az ilyen alkalmazásokra is; felismerik az általuk futtatott böngészőt és platformot, majd ezek alapján irányítják a kiszolgálást a megfelelő címre.

5.2. Index [6]

A tapasztalatok szerint az Index nem kezeli automatikusan a mobil klienseket, így azok lekérdezéskor a teljes tartalmat megkapják. Az oldal a felépítése miatt sok adatból áll, így azok letöltése is időigényes, valamint a megjelenítéshez szükséges memória méretigénye sem elhanyagolható ilyen mobil környezetben.

A tapasztalataink szerint a Nokia N91-es telefon jelezte is, hogy nincs elegendő rendelkezésre álló szabad memóriája, ezt azonban próbálhatjuk orvosolni azzal, hogy néhány futó alkalmazásból kilépünk.

5.3. Origo [7]

Ez a honlap is hasonló méretekkel rendelkezik, mint az előbb bemutatott, ám a fejlesztők itt már gondoltak a mobil kliensekkel érkező látogatókra, így őket átirányítják a számukra kialakított oldalakra.

6. Összegzés

A tesztek során kiderült, hogy érdemes olyan telefont választani, amely már gyárilag olyan böngészővel érkezik, ami teljesen Ajax-kompatibilis (Nokia Browser, NetFront, Safari). Ha mégsem így döntenénk, akkor használhatjuk az Opera Mini-t is, ami a Java miatt talán lehet, hogy egy picit lassabb, mégis sokszor jobban teljesít, mint egyes előre telepített böngészők.

A kényelmesebb böngészés érdekében esetleg érdemes figyelni arra, hogy mely készülékeken forgatható el a kijelzőn az alkalmazás ablaka, mivel az olvasás során kényelmesebb hosszabb sorokat végigkövetnünk a szemünkkel.

Irodalom

[1] World Wide Web Consortium, Web Standards, 1994-2008. http://www.w3.org/
[2] Opera Software, Opera Mini, 2008. http://www.opera.com/mini/
[3] Mozilla Foundation, Mobile – MozillaWiki, 2008. https://wiki.mozilla.org/Mobile
[4] [origo] Zrt., iWiW, 2005-2008. http://iwiw.hu/
[5] [origo] Zrt., iWiW Mobil, 2005-2008. http://m.iwiw.hu/
[6] Index.hu Zrt., 1999-2008. http://index.hu/
[7] [origo] Zrt., Origo, http://origo.hu/