Debreceni Egyetem
Informatikai Kar

Webes űrlapok és az XForms ajánlás

Témavezető:
Dr. Adamkó Attila
egyetemi adjunktus

Készítette:
Hetei György
programtervező Informatikus

Debrecen
2010

Bevezetés

Egy interaktív weboldal nem létezhetne űrlapok nélkül. Az űrlap az a része az oldalnak, amin keresztül kommunikálhatunk a felhasználóval. A weben nagyon sok űrlappal találkozhatunk. Például amikor regisztráljuk magunkat egy oldalra, bejelentkezünk, keresünk vagy más hasonló tevékenységeket végzünk, akkor ezt mind űrlapokon keresztül tesszük.

Fejlesztői oldalról egy űrlap létrehozása egyszerű, csupán fel kell sorolni a részeit, meghatározni, hogy dolgozza fel az adatokat, és ehhez valamilyen stílust kell definiálni. Ennek az egyszerűségnek viszont ára van. Egy HTML űrlap nem tudja eldönteni, hogy a felhasználó által bevitt információ a megfelelő formában van-e megadva, és egyszerű számítások elvégzését sem tudja elvégezni. Erre megoldást a JavaScript nyújthat, amely képes az oldalon belül a mezők értékét manipulálni, validálni. De a scriptek megírása időigényes, és ha néhány hónap múlva valami miatt hozzá kell nyúlni a kódhoz, nem biztos, könnyen tudunk tájékozódni benne, ha meg nem is mi írtuk, akkor nem kis fejfájást okozhat a kód módosítása. Ráadásul a böngészőkben biztonsági okok miatt a JavaScript letiltható.

A HTML űrlap hiányosságait figyelembe véve a W3C létrehozta az XForms szabványt, ami a webes űrlapok következő generációja. Az XForms képes az adatok validálására, egyszerűbb számítások elvégzésére, így elkerülhetjük, hogy JavaScriptet kelljen használnunk. Az XForms deklaratív, ezért könnyebb elsajátítani és használni. A kód későbbi módosítása is sokkal egyszerűbb, mint a JavaScript-nél. Ráadásul elkülöníti az adatstruktúrát a megjelenítéstől, ezáltal az egyes részek újrafelhasználhatóak lesznek. Nagy előnye a HTML űrlapokkal szemben, hogy használhatunk beépített adattípusokat, vagy hozhatunk létre saját típust is.

Szakdolgozatom célja az XForms technológia előnyeinek ismertetése a HTML űrlapokkal szemben, és egy olyan alkalmazás elkészítése, amelyen keresztül bemutatom ezeket az előnyöket.

Jelenleg egyetlen böngésző sem támogatja közvetlen módon az XForms technológiát. Több lehetőségünk is van az XForms használatára. Az egyik lehetőség, hogy egy beépülő modult telepítünk a böngészőre, ami képes feldolgozni az XForms utasításokat. A másik megoldás egy olyan alkalmazás használata, ami szerver oldalon átalakítja az XForms-t tartalmazó XHTML dokumentumot egy szabványos HTML oldallá. Ezek az alkalmazások mind az Ajax technológián alapulnak, ami azzal jár, hogy az XForms által meghatározott működést

lecserélik JavaScriptre. Ezek kicsit lassabbak, de minden JavaScriptet ismerő böngésző képes ezeket megjeleníteni.

Az alkalmazásom megjelenítésére az első módszert választottam, pontosabban a Mozilla Firefox böngészőjéhez készített kiegészítőt fogom használni. Igaz vannak még hiányosságai, de ez a legegyszerűbb megoldás az XForms használatára, ráadásul ingyenes.

A készített űrlapok által összegyűjtött adatokat a Java Servlet technológia segítségével fogom feldolgozni. Röviden a servlet egy olyan java objektum, amely http kérést dolgoz fel és http választ generál. A generált tartalom lehet HTML vagy akár XML is. A servletek kezelését egy servlet tárolóval rendelkező webszerver végzi. Ilyen az általam is használt Apache Tomcat. Az adatokat Mysql adatbázisban fogom tárolni.

A HTML űrlap

Az űrlap a HTML oldalaknak egy igen gyakran használt része, ugyanis ezen keresztül kérhetünk be információkat a felhasználótól. A HTML űrlap definiálására a *form* címke szolgál. Ez az elem írja le az űrlap néhány fontos aspektusát, és megmondja azt is, hogy hova és milyen módon küldjük az űrlapot.

Űrlap létrehozása

```
<form action=”_URL_” method=”get|post”>
…
</form>
```

Az *action="_URL_"* meghatározza az űrlapot feldolgozó program elérési útját. A *method="get|post"* azt mondja meg, hogy hogyan küldje az adatokat a feldolgozó programnak. A GET metódus a lap URL-jében küldi tovább az adatokat, ezzel szemben a POST metódus az űrlap szkriptjén keresztül továbbítja az adatokat.

Az űrlapok strukturált adatok cseréjét valósítják meg. A HTML űrlapoknál a struktúra név és érték párok halmazából áll. Az XForms ezzel szemben XML-ben küldi az adatokat, így bonyolultabb struktúrát is létre lehet hozni, ami nagyban megkönnyítheti az adatok feldolgozását.

A HTML űrlapok vezérlő elemei

A felhasználóval történő kommunikáció vezérlőkkel történik. A legtöbb vezérlőnek kötelező megadni a *name* attribútumát, mert ezzel lehet megfogni az adott vezérlő elemet.

Minden vezérlőnek van egy kezdő- és egy aktuális értéke, mindkettő karaktersorozat. A vezérlő aktuális értéke először felveszi a kezdőértéket, és ezután már a felhasználó, vagy egy script módosítja. Az XForms viszont szétválasztja az adatokat a vezérlőktől, ezáltal átláthatóbb és könnyebben módosítható struktúrát hoz létre.

Gombok

A gomboknak 3 fajtája van

- Submit gomb: aktiválásakor elküldi az űrlapot, egy űrlap több ilyen gombot is tartalmazhat
- Reset gomb: Az űrlap minden vezérlőjét visszaállítja a kezdő állapotba

- Nyomógomb: Nincs alapértelmezett viselkedésük, valamilyen kliens oldali szkriptet hívnak meg

Létrehozásuk történhet az *input* vagy a *button* címkével is.

```
<input type=”submit” value=”Submit gomb”>
<input type=”reset” value=”reset gomb”>
<input type=”button” value=”szkript” onclick=”szkript_nev()”>
```

Checkbox

Ennek a mezőnek két állapota lehet (be/ki). Egy checkbox bekapcsolt állapotban van, ha a *checked* attribútuma be van állítva. Több checkbox ugyanabban az űrlapban osztozhat ugyanazon a néven.

```
<input type=”checkbox” name=”elfogadas”>
```

Rádio gombok

Hasonló a checkboxhoz, de ha több rádió gombnak azonos a neve egy űrlapon belül, akkor csak az egyik lehet bekapcsolt állapotban.

```
<input type=”radio” name=”nem” value=”Férfi”>
<input type=”radio” name=”nem” value=”Nő”>
```

Menük

A menük több lehetőséget ajánlanak fel a felhasználónak, amik közül választhat.

```
<select name=”kivalasztas”>
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
</select>
```

Szöveg bevitel

Két eszközünk van adat bekérésére a felhasználótól. Az egyik az *input*, amivel egysoros szöveget kérhetünk be, a másik a *textarea,* ami többsoros szöveg begépelését teszi lehetővé. Mindkét esetben a bevitt szöveg lesz a vezérlő aktuális értéke.

Egysoros szöveg beviteli módok

Egysoros szöveg bevitelére alkalmas mezőt lehet létrehozni.

```
<input type=”text” name=”input” value=”Egysoros szöveg”
```

Érzékeny adatok kezelésére használható a jelszó típusú mező, ami a bevitt szöveg karaktereit lecseréli egy másik karakterre a megjelenítésnél. Ezzel az eszközzel megakadályozhatjuk, hogy a felhasználó képernyőjét nézők elolvashassák a felhasználó bizalmas adatait, mint például a jelszó.

```
<input type=”password” name=”jelszo”>
```

Ha több soros szöveget várunk a felhasználótól, akkor használhatjuk a *textarea* címkét.

```
<textarea name=”szövegdoboz”>
```

Fájl kiválasztás

Ez a vezérlő lehető teszi, hogy a felhasználó egy fájl tartalmát küldje el.

```
<input type=”file” name=”fajl”>
```

Rejtett vezérlők

Lehetőség van olyan vezérlőket létrehozni, amik nem jelennek meg az oldalon, de értékük továbbítódik. Ezekkel valamilyen plusz információt továbbíthatunk a szerver felé. Az XForms-ban már nincs rejtett mező, de az adatpéldányokban akárhány kiegészítő adatot elhelyezhetünk.

```
<input type=”hidden” name=”rejtett”>
```

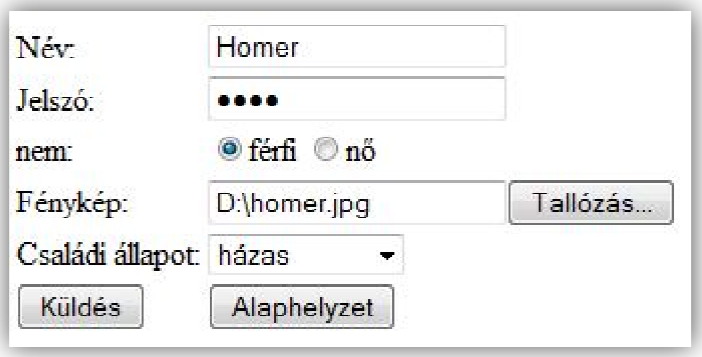
Egy HTML űrlap

A lenti kód egy egyszerűbb űrlapot definiál.

```
<form action="reg" method="get" >
  <table>
    <tr><td>Név:</td><td><input type="text" name="nev"></td></tr>
    <tr><td>Jelszó:</td><td><input type="password" name="jelszo"></td></tr>
    <tr><td>nem:</td><td><input type="radio" name="nem" value="férfi">férfi
                         <input type="radio" name="nem" value="nő">nő</td></tr>
    <tr><td>Fénykép: </td><td><input type="file" name="kep"></td></tr>
    <tr><td>Családi állapot:</td><td><select name=csaladi_allapot>
                                    <option>egyedülálló</option>
                                    <option>házas</option>
                                    <option>elvált</option>
                                    <option>özvegy</option>
                                   </select>          </td></tr>
    <tr><td><input type="submit"></td><td><input type="reset"></td></tr>
 </table>
</form>
```

1. ábra: Egy HTML űrlap

Ennek eredményeképp létrejön egy űrlap, ami az alábbi módon jelenik meg a böngészőben:



2. ábra

A küldés a GET metódussal történik ezért a laphoz tartozó URL tartalmazni fogja az űrlap elemeit. Az első része tartalmazza az *action* attribútumban megadott URL-t, és egy kérdőjel után szerepelnek a mezők „név=érték” formátumban megadva, egymástól & jellel elválasztva. Az űrlapon a jelszót egy „password” típusú vezérlőn keresztül kértük be, aminek segítségével elrejthettük a mező tartalmát, de küldésnél már egyszerű karaktersorozatnak tekinti az űrlap, ezért az URL-ben úgy látható, ahogy a felhasználó beírta.

```
                                                    apot=h%E1zas
```

A HTML űrlap korlátai és az XForms előnyei

A HTML űrlap egyik problémája a script nyelvektől való függőség. Nincs lehetőség a felvitt adatok validációjára kliens oldalon, és egyszerűbb számításokat sem tud elvégezni anélkül, hogy a szerverhez fordulna. Az XForms használatával jelentősen csökkenthetjük a scriptek számát, mert egy keretrendszert definiál, az egyszerűbb XPath alapú számítások és érvényesítések elvégzésére. Lehetőséget nyújt arra, hogy automatikusan értesíthetsük a felhasználót, ha érvénytelen adatot írt be, dinamikusan létrehozhatunk táblázatokat.

Másik gond a HTML űrlapokkal, hogy az adatokat név-érték formátumban küldi el, ezzel szemben az XForms az adatokat XML-ben küldi.

XML technológiák

Az XForms használatához meg kell ismernünk néhány XML technológiát, mint például magát az XML-t, mint dokumentumot, az XML Schema-t ami egy sémanyelv és az XML dokumentumok érvényességét ellenőrzi, és az XPath-t amivel egy XML dokumentumon belül tudunk navigálni. Érdemes megismerkedni a CSS stílusleíró nyelvvel, mert vele egyszerűen tudjuk szabályozni az oldalunk kinézetét

XML

Az XML (Extensible Markup Language, Kiterjeszthető Jelölő Nyelv) dokumentum olyan szöveges objektum, amely a szabvány előírásai szerint jól formáltak. Fizikailag egy olyan dokumentum, ami entitásnak nevezett tárolási egységekből, logikailag elemekből, megjegyzésekből, és egyéb szerkezeti elemekből áll.

Egy XML dokumentum jól formált, ha teljesíti az alábbi követelményeket:

- Egy dokumentumban egyetlen gyökérelem lehet.
- Minden nyitó címkéhez tartozik egy záró címke, de üres elem esetén a nyitó címke lehet záró címke is (<*ÜresElem*/>)
- Minden attribútum érték idézőjelek között áll. Használható a szimpla és a dupla idézőjel is, de attribútum-értékenként csak az egyik.
- A címkék egymásba ágyazhatók, de nem lehetnek átfedők. Mindegyik nem gyökér elemet másik elemnek kell magában foglalnia.
- A dokumentum megfelel a karakterkészlet definíciónak.

Egy XML dokumentum érvényes, ha teljesíti a hozzá tartozó XML séma által tartalmazott korlátozásokat

XML Schema

Az XForms az adatok érvényességét egy sémanyelv az XML Schema segítségével végzi. Egy XML séma az XML dokumentum típusának leírására, jellemzően az XML megkötésein túli korlátozásokat tartalmaz az adott típus struktúrájára és tartalmára. Az XML Schema sokoldalú adattípus rendszert használ, ami részletesebb megkötéseket tesz lehetővé az XML dokumentum logikai szintjén, de ezért sokkal robusztusabb érvényesítő keretrendszert követel

meg. XML alapú ezért szokványos XML eszközöket lehet használni a létrehozásához és feldolgozásához.

Az adattípusok leírásánál az XML Schema különbséget tesz lexikális tér és érték tér között. A lexikális tér azt mondja meg, hogy az adat hogyan jelenik meg az XML dokumentumban, Az érték tér pedig az adat megjelenését írja le absztrakt szinten. A gyakorlatban sok adattípus rendelkezik egy az egyhez kapcsolattal a lexikális és az érték tér között, így a megkülönböztetés csak elméletinek tűnhet. Például a logikai adattípus igaz értéke reprezentálható az ’1’ számmal vagy a ’true’ szöveggel. Ez fontos akkor mikor két érték egyenlőségét vizsgáljuk. Egyenlőség vizsgálatnál az értékek összehasonlítási alapja az értéktér.

Lehetséges egy létező adattípust átalakítani pontosan olyanra, mint amire szükségünk van. Ezt nevezik korlátozással történő származtatásnak. Például a következő XML Schema rész létrehoz egy új adattípust, ami lekorlátozza a sztring adattípus hosszát 3 és 15 karakter közé.

```
<xs:simpleType name="címkeTípus">
    <xs:restriction  base="xs:string">
        <xs:minLength value="3"/>
        <xs:maxLength value="15"/>
    </xs:restriction>
</xs:simpleType>
```

Több szempont szerint is lehet korlátozni az adattípusokat

- *enumeration* – meghatározza a lehetséges értékeket
- *fracionDigits* – meghatározza a tizedesvessző után álló számjegyek számát
- *length* – meghatározza a karakterek vagy a bájtok pontos hosszát.
- *maxExclusive* – meghatározza azt a maximum értéket, amit már nem vehet fel
- *maxInclusive* – a maximális értéket határozza meg
- *minExlusive* – meghatározza a minimum értéket, ami fölött felveheti az értékeket
- *pattern* – meghatároz egy reguláris kifejezést a lexikális térrel szemben
- *totalDigits* – meghatározza a számjegyek számát

- *whiteSpace* – meghatározza, hogy hogyan kell kezelni a *whitespace* karaktereket

Az egyik leghasznosabb facet alapú korlátozás az űrlapoknál a *pattern*, amely reguláris kifejezéseket használ. Például ha egy telefonszámot 123-456 formában szeretnénk bekérni, akkor annak ellenőrzésére, hogy a felhasználó a megfelelő formátumban adta meg a számát a *'[1-9]{3}-[0-9]{3}'* reguláris kifejezést kell használnunk.

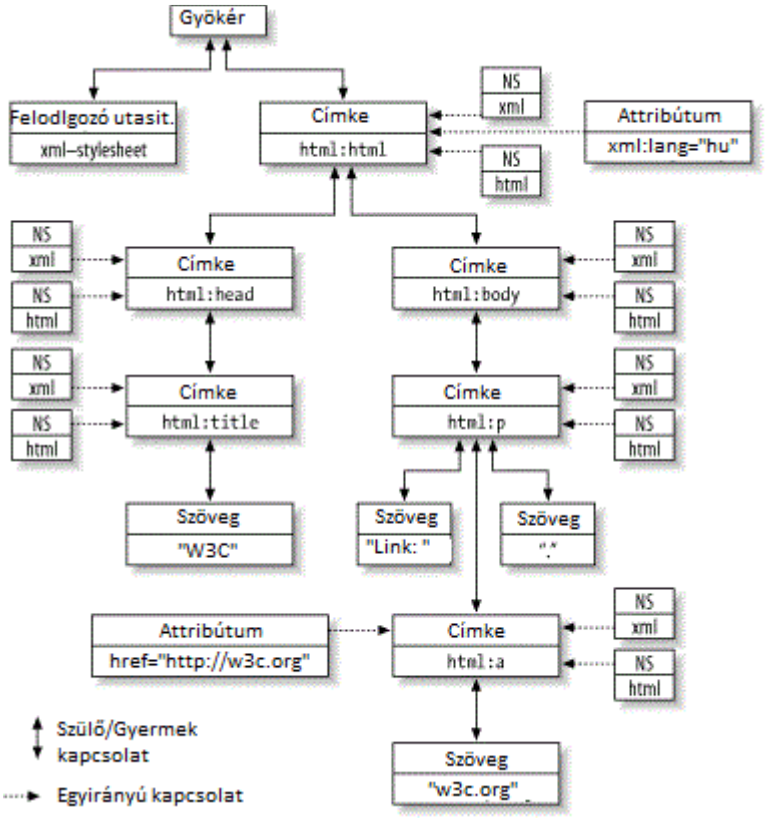
XPath (XML Path Language)

Mivel az XForms szétválasztja az adatmodellt a megjelenítéstől, ezért ezt a két részt valamilyen módon össze kell kapcsolnunk. Erre nagyon jó eszköz az XPath, ami egy nyelv, amely XML csomópontok kiválasztására szolgál egy XML dokumentumból. Ezen kívül az XPath használható értékek kiszámítására egy XML dokumentum tartalma alapján. Az XPath egy XML dokumentum fa alapú reprezentációján alapszik.

Az XPath adatmodell

Az XPath-ban az elemek, attribútumok, szövegek, megjegyzések, feldolgozó utasítások és még a névterek is egy fa csomópontjaiként vannak ábrázolva. Néhány csomópontnak, mint például a címkének lehet gyereke, míg másoknak, mint például az attribútumoknak nincs. A fának van egy kitűntetett csomópontja, a gyökér.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<?xml-stylesheet href="screen.css" type="text/css" media="screen"?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="hu">
  <head>
    <title>W3C</title>
  </head>
  <body>
    <p>Link: <a href="http://w3c.org/">w3c.org</a>.</p>
  </body>
</html>
```



3. ábra: XPath adatmodell példa

A fenti példán látható, hogy sem az XML sem a DOCTYPE deklaráció nem jelenik meg a csomópontok között. Ez azért van, mert ezek az XML adat struktúrák tulajdonképpen láthatatlanok az XPath számára. Továbbá látható, hogy minden címke csomóponthoz két névtér (NS) csomópont kapcsolódik. Egyik a gyökér címke *xmlns:html* deklarációja, amely hatóköre az egész dokumentum, a másik az *xml* előtag, ami egy beépített deklaráció ami *xml:lang* néven látszik. Ezen a példán látható, hogy még egy ilyen kisméretű dokumentum is, mint ez milyen sok csomópontot generál.

Minden csomópont rendelkezik tulajdonságok halmazával, amelyeket az XPath különböző módokon tár fel. Ezek a tulajdonságok a következők:

- Név. Néhány csomópontnak, mint például a gyökérnek, a megjegyzéseknek, a szöveg csomópontoknak mindig van nevük. Címkék és attribútumok egy kiterjesztett névvel rendelkeznek, ami egy helyi névnek és a névtér URI-jának kombinációjából áll. A feldolgozói utasítások a céljukat használják névként. A névtér csomópontok pedig a névtér előtagot használják névként.

- Sztring érték. A szöveges csomópontok az XML dokumentumból tartalmaznak karaktereket. A megjegyzés csomópontok a megjegyzés teljes szövegét tartalmazzák, amibe nem tartozik bele a megjegyzés jelölésére vonatkozó eszköz. Az Attribútumok az attribútum értekeit tartalmazzák. A névtér csomópontok sztring értéke az URI vagy az üres sztring. A gyökér és a címke csomópontok a rekurzívan számítják ki ezt a tulajdonságot a leszármazott csomópontok sztring értékei segítségével.
- Gyermek. Elméletileg minden csomópontnak lekérdezhető a gyermeke, de csak a gyökér és a címke csomópontok fognak rendelkezni vele.
- Pozíció, ami relatív érték a többi csomóponthoz képest.

Ezeken kívül néhány csomópont rendelkezhet az alábbi tulajdonságokkal is:

- Szülő. Minden csomópont rendelkezik vele, kivéve a gyökér csomópontot.
- Attribútumok. Címkék is tartalmazhatnak attribútumokat, amelyeket speciálisan kell kezelni, és nem nincs gyermekük
- Névterek. A névtér csomópontok is speciálisan kezelendőek.

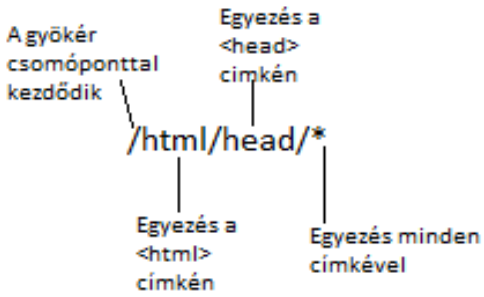
A csomópontok gyűjteményét, amelyben nem fordul elő ugyanaz a csomópont egynél többször, csomópont halmaznak nevezzük.

Hozzáférési utak

Hozzáférési utak segítségével tudunk megfogni egy vagy több csomópontot a fában lévő helyük vagy más tulajdonságuk alapján. Egy hozzáférési út számos egyéni hozzáférési lépésből áll, amik egy per (/) karakterrel vannak elválasztva. Minden egyéni lépés az előző lépés alapján épül fel, ahogy áthalad a dokumentumon, és tesztelhető csomópont neve ellen vagy a következő speciális tesztekkel:

- node() – Az összes csomópontra egyezik.
- text() – Az összes szöveges csomópontra egyezik.
- comment() – Egyezik az összes megjegyzés csomópontra.
- processing-instruction() - Egyezik bármely feldolgozói utasítás csomópontra, ezen kívül megadható egy paraméter, amivel speciális feldolgozói utasításokra is egyezést ad.

- * - egyezni fog az összes címke csomópontra vagy attribútum csomópontra, vagy a névtér csomópontra a névtér tengelyen belül.



4. ábra: hozzáférési út példa

Kontextus

A hozzáférési utak egy nagyobb XML dokumentumban elég hosszúak lehetnek, ezért az XPath-ban lehetőségünk van egy aktuális csomópont megadására. Ehhez a csomóponthoz képest relatívan tudunk hivatkozni más csomópontokra. Az aktuális csomópont az alábbi elemekből áll:

- Egy aktuális csomópont
- egy nem nulla pozitív egész számpár (a kontextus helye és mérete).
- Változó összerendelések
- Egy függvénykönyvtár
- névtér deklarációk halmaza

Egy „.„–ból (pont) álló kifejezés kiválasztja az aktuális csomópontot, a „..”-ból álló az aktuális csomópont szülő csomópontját választja ki. Azokat a kifejezések, amelyek perjellel kezdődnek abszolút utaknak nevezzük, amik függetlenek az aktuális csomóponttól

Tengelyek (axes)

Minden hozzáférési lépés tartalmaz egy tengelyt, amely egy utasítás, ami megmondja, hogy hogyan kell navigálni a fa-struktúrában. Az alábbi tengelyek léteznek:

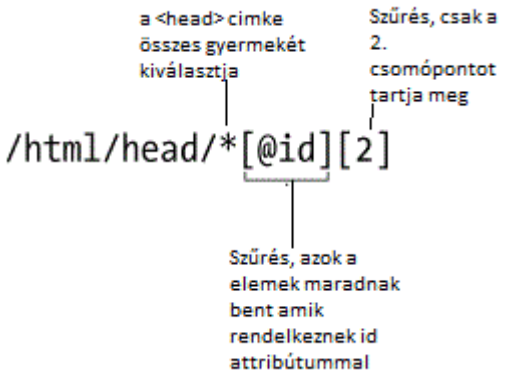
- child: gyerek

- descendant: leszármazott
- parent: szülő
- ancestor: ős
- following-sibling: következő testvér (ugyanazon a szinten lévő elem)
- preceding-sibling: előző testvér (ugyanazon a szinten lévő elem)
- following: következő csomópont
- preceding: előző csomópont
- attribute: attribútum
- namespace: névtér
- self: önmaga
- descendant-or-self: leszármazott vagy önmaga
- ancestor-or-self: ős vagy önmaga

Predikátumok

Egy egyszerű hozzáférési út kiválaszt minden csomópontot, ami egyezik az őt leíró úttal. Például a */filmek/film/* kifejezés kiválaszt minden *film* címkét, ami a *filmek* címkének közvetlen gyermeke. De általában szükségünk van arra, hogy a kiválasztás eredményét valamilyen módon szűrjük. Az XPath-ban lehetőség van rá predikátumokon keresztül. A predikátumok szögletes zárójelben szerepelhetnek a hozzáférési út bármely hozzáférési lépésében. Például, ha csak az első *film* címkét akarjuk kiválasztani, akkor a */filmek/film[position( ) = 1]* kifejezést vagy ennek rövidebb változatát */filmek/film[1]* kell használnunk.

Ha a hozzáférési lépéshez tartozó predikátum kifejezés értéke igaz, akkor a csomópont bekerül az eredmény halmazba. Minden lépéshez több predikátumot is hozzá lehet rendelni. Ha egy csomópont halmaz a szűrések után üres lesz, akkor a kifejezés egy üres csomóponttal ad eredményül.



5. ábra: példa a predikátumok használatára

Számított érték

Az XPath képes elvégezni egyszerűbb számításokat, aminek eredménye lehet sztring, logikai, vagy szám típusú. Az XPath kifejezéseknek négy fontosabb adattípust ismernek, ezek a csomópont-halmaz, sztring, logikai és a szám típus.

Operátorok

Az XPath operátorok precedenciája a leggyengébbtől a legerősebbig:

- or
- and
- =; !=
- <=; <; >=; >
- +; -
- *; div; mod; unary -
- | (unió)

Kifejezés kiértékelésénél az azonos szinten lévő operátorokat balról jobbra sorrendben dolgozza fel. A kifejezés kiértékelésének sorrendjét zárójelek segítségével tudjuk módosítani.

Függvények

Az XPath számos beépített függvénnyel rendelkezik, amelyek a közül kettőt mutatok be:

number **position**() – Az aktuális elem sorszáma a vele egy szinten lévők között

string **concat**(string, string, string?) – A paraméterben megadott sztringeket összefűzi

XHTML

Az XHTML a HTML megfogalmazása XML-ben. A két nyelv között annyi az eltérés, hogy az XHTML formai követelményei szigorúbbak:

- A DOCTYPE kivételével mindent kisbetűvel kell írni.
- Minden elemet le kell zárni. Az üres elemeket önmagukban egy szóközzel és egy / jellel:
- Az elemeket csak egymásba ágyazva lehet használni.
- A jellemzőket idézőjelek közé kell írni!
- A jellemzőknek legyen értéke!

Az XHTML 1.0 2000. január 26-án lett W3C ajánlás. AW3C 2006- bejelentette az XHTML 2.0 munkatervét. A tervben szerepel az XForms használata a webes űrlapok helyett, továbbá az XFrames, XML események alkalmazása. De 2009-ben a W3C leállította a fejlesztését, hogy a készülő HTML5-öt hamarabb befejezze.

CSS

A **CSS** (angolul ***C****ascading* ***S****tyle* ***S****heets*) a számítástechnikában egy stílusleíró nyelv, mely a HTML vagy XHTML típusú strukturált dokumentumok megjelenését írja le. Ezen kívül használható bármilyen XML alapú dokumentum stílusának leírására is, mint például az SVG, XUL stb.

A CSS-t a weblapok szerkesztői és olvasói egyaránt használhatják, hogy átállítsák vele a lapok színét, betűtípusait, elrendezését, és más megjelenéshez kapcsolódó elemeit. A tervezése során a legfontosabb szempont az volt, hogy elkülönítsék a dokumentumok struktúráját (melyet HTML vagy egy hasonló leíró nyelvben lehet megadni) a dokumentum megjelenésétől (melyet CSS-sel lehet megadni). Az ilyen elkülönítésnek több haszna is van, egyrészt növeli a weblapok használhatóságát, rugalmasságát és a megjelenés kezelhetőségét, másrészt csökkenti a dokumentum tartalmi struktúrájának komplexitását. A CSS ugyancsak alkalmas arra, hogy a dokumentum stílusát a megjelenítési módszer függvényében adja meg,

így elkülöníthető a dokumentum formája a képernyőn, nyomtatási lapon, hangos böngészőben, vagy braille készüléken megjelenítve.

Az elemek stílusát különböző CSS szelektorokkal lehet kiválasztani:

- *Minden elemre* – a `*` szelektor használatával
- *Az elem neve alapján* – például minden '`p`' vagy '`h2`' HTML-elemhez
- *Leszármazottak alapján* – például az olyan '`a`' elemekre, melyek egy '`li`' elem részei, a szelektor `"li a"`
- class *vagy* id *attribútumok alapján* – például *.class* és/vagy *#id* a `class="osztály"` vagy `id="azonosító"` elemekhez

Ezeken kívül rendelkezésre áll több *pszeudo-osztály,* melyekkel további műveletekhez lehet stílust rendelni. Talán a legismertebb ezek közül a `:hover`, melynek stílusa akkor lép érvénybe, mikor a hozzá tartozó elem aktiválódik, például fölé visszük az egeret. Hozzá lehet fűzni a szelektorokhoz is, például `a:hover` vagy `#elementid:hover`. További ismertebb pszeudo-osztályok a `:first-line`, a `:visited` vagy a `:before`.

Az XForms

Miért is jó az XForms?

- Az XForms elkülöníti az adatokat a megjelenítéstől.
- Lehetőséget biztosít kliens oldali validációra, és egyszerűbb számítások elvégzésére.
- Deklaratív, így könnyebb megtanulni, karbantartani és javítani.
- Tele van felhasználói felület vezérlőkkel az összetett adatok kezelésére, mint például a dátumok, számok és tartományok.
- Kompatibilis az XML szabványokkal, mint például az XML Schema és az XPath.
- Bővíthető

Az XForms modell

Az XHTML dokumentumon belül az XForms két külön részben jeleneik meg. Az első rész a modell, amely leírja, hogy milyen adatszerkezeteket fogunk használni, meghatározza az adatok típusát, kezdőértékét, és a feldolgozásuk módját is. A másik rész a felhasználói felület, amely az űrlap megjelenését határozza meg. A modell megadására a *model* címke szolgál. Tipikusan egy nem megjelenített részén helyezkedik el a dokumentumnak. Ha egy dokumentumban több űrlapot akarunk használni, akkor több modellt kell megadnunk. Egy modell egy vagy több adatpéldányt tartalmaz. Az adatpéldányon belül adhatjuk meg az adatszerkezetet.

```
<xf:model schema="film.xsd" id="model-újFilm">
 <xf:instance id="film/>
   <film>
    <azon>0</azon>
    <cím/>
    <leírás/>
    <szereplők/>
    <címkék/>
    <értékelés>1</értékelés>
    <megjelenés/>
    <borítóFájl/>
    <korhatár korhatáros="false">
        <év>12</év>
        <év>14</év>
        <év>16</év>
        <év>18</év>
    </korhatár>
   </film>
 </xf:instance>
 <xf:instance id="címkék">
     <címkék xmlns="">
        <címke szöveg=""/>
     </címkék>
 </xf:instance>
 <xf:instance id="kiválasztottCímke">
           <címke szöveg=""/>
 </xf:instance>
</xf:model>
```

Lehetőség van rá, hogy egy külső XML dokumentumra hivatkozzunk, ezzel csökkentve a kód méretét. Csak az adatpéldány létrehozására szolgáló címkét kell módosítanunk:

```
<xf:instance id="film" src="film.xml"/>
```

A modelleknek és az adatpéldányoknak megadható egy *id* attribútum, amivel azonosíthatjuk őket. Így később ezekkel az azonosítókkal hivatkozhatunk rájuk. Az adatok érvényességi vizsgálatához szükséges megszorításokat is a modellben kell megadnunk, ezt megtehetjük úgy, hogy egy XML Schema dokumentumot hozzákapcsolunk az adott modellhez.

A modellben használható a *bind* címke, egyedi azonosítót és a csomópont viselkedését meghatározó modell-elem tulajdonságokat rendelhetünk a csomópontokhoz. Az azonosítónak egyedinek kell lennie a dokumentumban, így a hivatkozásnál nem kell megadnunk, hogy melyik modell melyik adatpéldányában van az adott csomópont. A modell-elem tulajdonságok különböző megszorításokat alkalmaznak a csomóponton. A tulajdonságok közül néhány XPath kifejezés, amelyeket az XForms feldolgozó figyeli és átértékeli, ha szükséges. Ezek a modell-elem tulajdonságok a következők:

- *readonly* – megjelöli, hogy a csomópont csak olvasható, így az űrlap vezérlő, amelyik alá van rendelve a csomópontnak nem fogja engedni hogy az adat megváltozzon. Értéke lehet bármilyen

- *required* – megjelöli a csomópontot, hogy az értékét szükséges megadnunk az űrlapban. A csomópont kielégíti a szükséges feltételt, ha éréke egy olyan sztringgé konvertálható, amely egy vagy több karakterből áll.

- *relevant* – megmondja, hogy a csomópont jelenleg fontos része-e az űrlapnak. Az űrlap vezérlők a nem releváns csomópontokat vagy kikapcsolt állapotban jelenítik meg, vagy meg sem jelenítik őket. A nem releváns csomópontok nem lesznek elküldve a többi adattal.

- *calculate* – egy számítást definiál, amely meghatározza a csomópont értékét. Értéke lehet bármilyen XPath kifejezés, aminek eredménye sztringként lesz értelmezve.

- *constraint* – ez a tulajdonság hozzáad még egy további XPath alapú megszorítást a hozzárendelt csomópont érvényességére vonatkozóan. Értéke lehet bármilyen XPath kifejezés, aminek eredménye logikai értékként lesz értelmezve.

A többi tulajdonság statikus és nem értékelődik át:

- *type* – összekapcsolja az adatpéldány csomópontot egy XML Schema–ban meghatározott adattípussal. Igazából ez a tulajdonság technikailag felesleges, mert egy jó XML Schema ugyanezt az eredményt megvalósítja

```
<xf:bind id="filmCím" nodeset="cím" type="xs:string"
required="true()" />
<xf:bind nodeset="/film/megjelenés" type="xs:date" />
<xf:bind nodeset="/film/értékelés" type="xs:integer" />
```

Továbbá a modellben kell megadnunk az adatok küldésére vonatkozó információkat. Erre a *submission* címke szolgál. Csak akkor küldi el az adatokat a szerver felé, ha az adatpéldányban megadott értékek eleget tesznek a séma által megadott megszorításoknak.

```
        <xf:submission id="filmekCímkeAlapján"
                       action="Filmek"
                       method="post"
                       encoding="UTF-8"
                       replace="instance"
                       ref="instance('kiválasztottCímke')"
                       instance="filmek"
                       >
        </xf:submission>
```

A fenti kód a *kiválasztottCímke* nevű adatpéldány tartalmát küldi el, és válaszul egy olyan XML dokumentumot vár, amely az adott címkével megjelölt filmeket tartalmazza. Ezt a műveletet az oldal újratöltése nélkül végre tudja hajtani. A fenti *submission* által küldött XML dokumentum lehet az alábbi:

```
<?xml version="1.0" encoding="UTF-8"?>
<címke szöveg="akció"/>
```

XPath az XForms-ban

Az XPath egy XML dokumentumban történő navigálást teszi lehetővé. XPath kifejezések segítségével megfoghatjuk a modellben lévő adatokat, és módosíthatjuk az értéküket például a felhasználó felület vezérlői segítségével.

Az XPath minden használata magában foglal egy aktuális csomópontot, általában azzal a céllal, hogy csökkentse a XPath kifejezések lépéseinek számát.

```
<film>
...
  <címkék>
     <címke szöveg=”cimke1”/>
     <címke szöveg=”cimke2”/>
     <címke szöveg=”cimke3”/>
  </címkék>
...
</film>
```

A fenti kódban az alapértelmezett kontextus csomópont a *film* nevű címke csomópont. Ezért hogy összekapcsoljunk egy űrlap vezérlőt a címkékkel ahelyett, hogy az abszolút utat használnánk *(/film/címkék/címke)*, használhatjuk a rövidebb relatív hozzáférési utat: *címkék/címke*. A fő különbség a kettő között az, hogy az abszolút út tartalmaz egy kezdő perjelet és a gyökércímke nevét. Mivel csak egy gyökércímke lehet, nem szükséges, hogy

minden hozzáférési út tartalmazza a nevét. Az XPath kifejezések akkor is egyértelműek maradnak, ha ezt a redundáns lépést kihagyjuk.

Az alapértelmezett aktuális csomópont megváltoztatható. Minden elem magába foglal egy összekapcsoló kifejezést. Az összekapcsoló kifejezések tartalmaznak egy *ref* attribútumot vagy egy *bind* attribútumot. Bármelyik módszert használjuk, a kifejezés kiválaszt egy csomópont halmazt és az első csomópont a dokumentumban lesz a kontextus csomópont a gyermekek számára.

Attribútumok

Számos helyzetben az XForms hivatkozik az adatpéldányra. Amikor az összekapcsoló attribútum célja, az hogy egyetlen csomópontot válasszon ki, akkor a *ref* attribútumot kell használnunk, ami egy XPath kifejezést tartalmaz. Azokban az esetekben ahol a kiválasztott csomópont halmaz egynél több csomópontot tartalmaz, akkor az első csomópont szabály alkalmazása történik, amely eltávolítja az összes csomópontot az első kivételével. A csomópontok sorrendjét a dokumentumban való megjelenésük sorrendje határozza meg.

Amikor egy csomópont halmazt akarunk kiválasztani az összekapcsolásra, akkor a *nodeset* attribútumot kell használnunk, ami szintén XPath kifejezést használ.

Az XForms felhasználói felülete

Az XForms felhasználói felületén keresztül történik a felhasználói interakció. A felhasználói felület vezérlőin keresztül a felhasználó módosítni tudja a modell tartalmát. Ezen kívül lehetőséget ad különböző műveletek elvégzésére.

Kapcsolat létrehozása

Az űrlap vezérlők a modellben szereplő adatokkal két módon léphetnek kapcsolatba. Az egyik módszer az IDREF alapú összekapcsolás. Ehhez a modellben minden egyes *bind* címkéhez meg kell adnunk egy *id* attribútumot, és erre az attribútumra kell visszahivatkoznunk minden űrlap vezérlőben.

```
<!—Az XForms modellben -->
<xf:bind id="filmCím" nodeset="cím" required=”true”/>
...
<!-- a felhasználói felületen -->
<xf:input bind="filmCím">
```

A legfőbb előnye, hogy elkülöníti a modellt és a nézetet. Így, ha az adatpéldány struktúrája megváltozik, csak a *bind* címkében lévő XPath kifejezést kell frissítenünk.

Másik lehetőség az összekapcsolásra XPath kifejezések használata az űrlap vezérlőkben.

```
<xf:input ref="/film/cím ">
```

A különbség, az hogy itt az űrlap vezérlőben a *ref* attribútumon keresztül történik az összekapcsolás. Sok szemszögből ez a megközelítés egyszerűbb, mivel egy szintet levág a hivatkozásból. Ha az adatpéldány struktúrája változna, akkor a *bind* címke attribútumát és az űrlap vezérlő attribútumát is meg kell változtatnunk.

Általános hogy ugyanaz a dokumentum több űrlapot tartalmaz és ez által több XForms modellt is.

```
<!-- Az 1. XFroms modell -->
<xform:model id="m1">
  <xforms:bind nodeset="email" type="my:email"/>
  ...
</xforms:model>
<!-- A 2. XFroms modell -->
<xforms:model id="m2">
  <xforms:bind nodeset="search" type="my:query"/>
  ...
</xforms:model>

<!-- később a dokumentumban -->
<xforms:input ref="email" model="m1"...>
<xforms:input ref="search" model="m2"...>
```

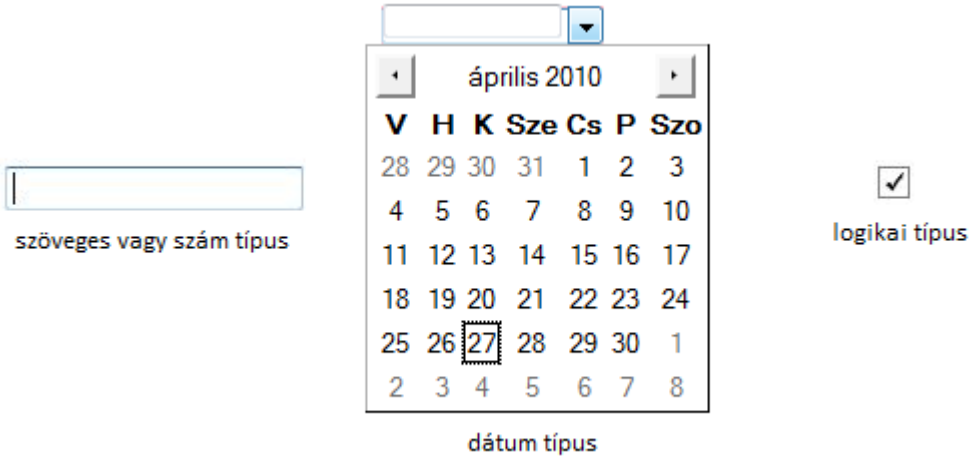
Ha az IDREF alapú összekapcsolást használjuk, nem okoz problémát, mivel az azonosító szükségszerűen egyedi az egész dokumentumban. De, ha a *ref* attribútummal kapcsolunk össze, további információkra van szükségünk, arra vonatkozóan, hogy az XPath kifejezés melyik modellhez tartozik.

Űrlap vezérlők

A felhasználóval történő kommunikációért az űrlap vezérlők a felelősek. Mindegyik vezérlőnek megadható egy *label* címke, amivel az űrlapon elnevezhetjük az adott elemet. Ez a név a vezérlők többségénél a vezérlőtől balra jelenik meg, de gombok esetében magán a gombon. A vezérlők megjelenése függ a hozzárendelt érték típusától, ezeket majd vezérlőnként bemutatom.

input – lehetővé teszi bármilyen karakteres adat beírását.

```
<input ref="film/cím"/>
```



6. ábra: az input vezérlő lehetséges megjelítései

secret – ennek a vezérlőnek a szerepe azonos a HTML űrlap *password* típusú vezérlőjével

```
<secret ref="jelszó"/>
```

textarea - többsoros szöveg bevitelét teszi lehetővé

```
<textarea ref="tartalom"/>
```

output – ez az egyetlen vezérlő, ami nem a felhasználótól érkező adatok fogadására szolgál. Az XForms modellből írja ki a hozzárendelt adatot a képernyőre. Szöveges típus esetén megkülönböztethetetlen a szöveges tartalomtól. Megadható neki egy hiperhivatkozás, egy kép vagy HTML típusú adat

```
<output ref="/film/cím"/>
```

upload – a HTML űrlapok is rendelkeztek egy korlátozott fájl feltöltő vezérlővel, de az XForms vezérlője több módon is felülmúlja.

```
<upload ref=”instance(’segéd’)”>
  <label>Válassz egy fájlt</label>
  <mediatype bind="/image/*"/>
</upload>
```

7. ábra: az upload vezérlő alapértelmezett megjelenése

Az XForms specifikáció több lehetséges forrást is megnevez a feltöltéshez:

- fájl feltöltés – általánosan alkalmazható és egyaránt működik képpel, hanggal, szöveggel, adattal és egyéb fájlokkal.
- firkáló - képekhez használható. Egyik módja hogy képeket generáljunk és kirajzoljuk.
- megszerzett kép – Ha lehetséges az XForms támogatja a fényképezőgépeket és a szkennereket, ahol az *output* űrlap vezérlő képet igényel.
- hang felvétele
- video felvétele

range – Ez az űrlap vezérlő nincs a HTML űrlapok vezérlői között. Lehetővé teszi, hogy egy tartományból válasszunk ki egy értéket. Az alsó és felső határokat a *start* és az *end* attribútumokkal lehet megadni, a lépésközt pedig a *step* attribútummal. Sajnos ezt a hasznos vezérlőt a Firefox kiegészítő jelenleg nem támogatja. Igaz megjeleníti, de az adatpéldány értékét nem képes megváltoztatni.

```
<xf:range model="model-újFilm" start="1" end="5" step="1"
ref="/film/értékelés" incremental="true">
```

8. ábra

trigger – Ez az űrlap vezérlő hasonlít a HTML űrlap gombjához, de képes több műveletet is végrehajtani egyszerre.

```
<xf:trigger appearance="minimal">
            <xf:label>új felvitele</xf:label>
            <xf:toggle ev:event="DOMActivate" case="case-újFilm"/>
</xf:trigger>
```

submit – ez az űrlap vezérlő egy speciális változata a *trigger*-nek, kiegészítve azt az űrlap küldésével.

```
<submit submission="adat">
  <label>Bejelentkezés</label>
</submit>
```

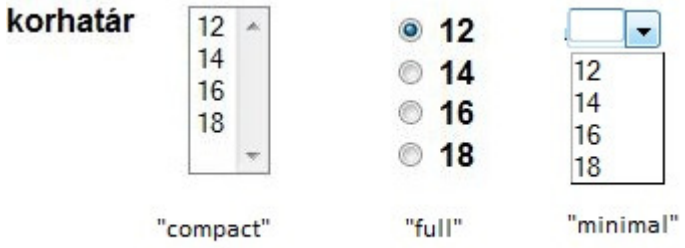
select1 – egy listából pontosan egy elem kiválasztását szolgálja. Amikor a kezdő adatpéldány tartalmaz egy értéket, ami lehet üres is, ami nem szerepel a *value* címkék között, akkor nem látható, hogy mi a kiválasztott adat. Minden egyes alkalommal, amikor új értéket választunk ki, módosulni fog az adatpéldány is. Egy speciális lehetőség ennek az űrlap vezérlőnek, hogy amikor a *selection* attribútuma *open* értékkel szerepel. Ez azt jelzi, hogy a szabad bevitel lehetséges, így a felhasználó egy tetszőleges értéket vihet be. Az egyik megadási módja mikor a *select1* címkén belül felsoroljuk az elemeket és azokon belül a név-érték párokat.

```
<select1 ref="/film/korhatár">
  <label>korhatár:</label>
  <item>
    <label>12</label>
    <value>12</value>
  </item>
   ...
  <item>
    <label>18</label>
    <value>18</value>
  </item>
</select1>
```

Ez a megoldás nem választja el az adatokat a megjelenéstől ezért érdemes kiemelni az opciókat a modellbe és az *itemset* címke segítségével hivatkozni rá.

```
<!-- az adatpéldány-->
<film>
...
    <korhatár>
        <év>12</év>
        <év>14</év>
        <év>16</év>
        <év>18</év>
    </korhatár>
...
</film>
...
<xf:select1 ref="korhatár" appearance="compact" >
    <xf:label>korhatár</xf:label>
    <xf:itemset nodeset="instance('film')/korhatár/év">
        <xf:label ref="."></xf:label>
        <xf:value ref="."></xf:value>
    </xf:itemset>
</xf:select1>
```

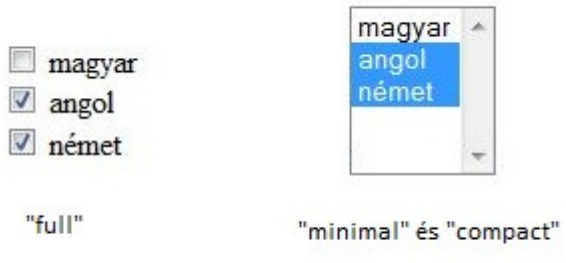
Ha később változtatni akarunk a korhatárokon, akkor elég csak az adatpéldányt módosítanunk. A megjelenés az *appearance* attribútumtól függ.



9. ábra: A select1 vezérlő *compact, full* és *minimal appearance* beállítással

select – listából nulla, egy vagy több dolog kiválasztását teszi lehetővé.

```
<xforms:select ref="nyelvek" appearance="minimal" >
    <xforms:itemset nodeset="instance('film')/nyelvek/nyelv">
        <xforms:label ref="."></xforms:label>
        <xforms:value ref="."></xforms:value>
    </xforms:itemset>
</xforms:select>
```



10. ábra: select vezérlő *full, minimal* és *compact* beállítással

Kapcsolat az adatpéldánnyal

Az űrlap vezérlők átalakítják a felhasználó által megadott adatokat karakterláncokká vagy gyermek elemekké. Az XForms a kapcsolatot XML-en keresztül tartja.

Amikor összekötünk egy attribútumot vagy egy címkét egy szöveges tartalommal, akkor az összerendelés egyértelmű, vagy az attribútum értéket vagy az attribútum szöveges gyermek csomópontját használva.

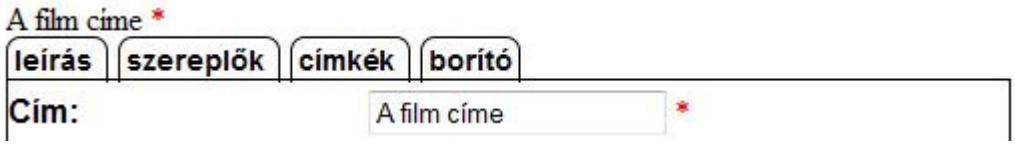
```
ref="/element"
<element>form data</element>

ref="/element/text(  )"
<element>form data</element> (egyenértékű az előzővel)

ref="/element/@attribute"
<element attribute="form data"/>
```

Azt, hogy mikor történik az adatpéldány és az űrlap vezérlő közötti szinkronizáció, egy attribútum mondja meg, az *incremental*. Ha ennek értéke *false*, akkor csak a felhasználó továbbnavigálása után történik meg az adatok frissítése. Ez az alapértelmezett beállítás az *input, secret, textarea, range* és az *upload* vezérlőknél. De ha az attribútum értéke *true*, akkor a frissítés gyakrabban történik meg.

```
<!-- a modell -->
<xf:instance>
...
<film>
    <cím/>
</film>
...
</xf:instance>
<xf:bind nodeset=”/film/cím” id=”filmcím”/>
<!-- a megjelenítés -->
...
<xf:output bind="filmCím"/>
...
<xf:input bind="filmCím" incremental="true">
     <xf:label>Cím: </xf:label>
</xf:input>
...
```



11. ábra: *incremental*

Az adatpéldány értéke mindig az input mező értékével fog megegyezni. Amikor változtatom a film címét a vezérlőben, akkor az változik az adatpéldányban, és mivel egy output vezérlő is ugyanarra az értékre hivatkozik, annak az értéke is megváltozik.

Csoportosítás

A csoportok számos kényelmes lehetőséget nyújtanak, de van funkcionális nézőpontjuk is. Egy csoport egy másik fajtája az űrlap vezérlőknek, és ezért a modell-elem tulajdonságok, mint például a *relevant* és a *requied* alkalmazható a csoportra, felülírva a tartalmazott űrlap vezérlők tulajdonságait. A leghasznosabb akkor, amikor egy egész részét egy űrlapnak meg kell változtatni az adatpéldány részben.

A *group* címke kényelmes hely lehet az XForms névtér, mint alapértelmezett névtér deklarálására, hogy csökkentse az ismétlődő prefixek és további deklarációk zűrzavarát.

Dinamikus megjelenítés

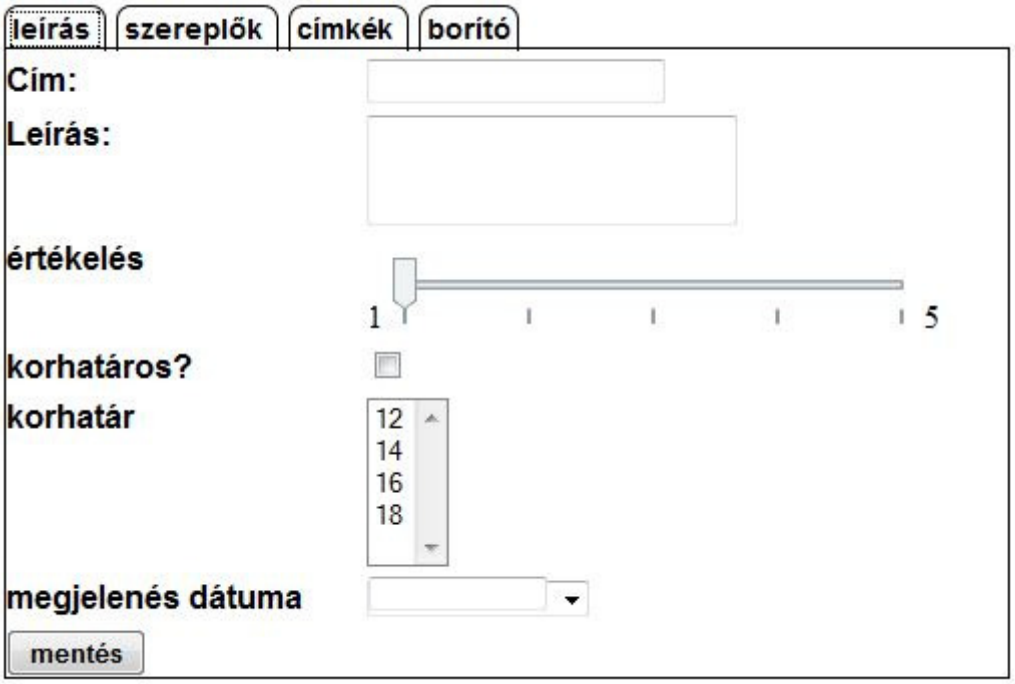
A *switch* címke egy tárolóként szolgál általában kettő vagy több *case* címkének. Adott pillanatban pontosan egy *case* van jelen a dokumentumban és az összes többi nem jelenik meg. Jó példa erre egy lapfüles felület.

```
<!-- a navigáláshoz szükséges triggerek -->
<xf:trigger appearance="minimal">
   <xf:label>leírás</xf:label>
   <xf:toggle ev:event="DOMActivate" case="t-leírás"/>
</xf:trigger>
   <xf:trigger appearance="minimal">
   <xf:label>szereplők</xf:label>
   <xf:toggle ev:event="DOMActivate" case="t-szereplők"/>
</xf:trigger>
<xf:trigger appearance="minimal">
    <xf:label>címkék</xf:label>
    <xf:toggle ev:event="DOMActivate" case="t-címkék"/>
</xf:trigger>
<xf:trigger appearance="minimal">
    <xf:label>borító</xf:label>
    <xf:toggle ev:event="DOMActivate" case="t-borító"/>
</xf:trigger>

<!-- a switch-case szerkezet -->

<xf:switch>
   <xf:case id="t-leírás">
     ...
   </xf:case>
   <xf:case  id="t-szereplők">
     ...
   </xf:case>
   <xf:case  id="t-címkék">
     ...
   </xf:case>
   <xf:case  id="t-borító">
     ...
   </xf:case>
</xf:switch>
```

A navigálást triggerek segítségével végzi. Minden triggerben szerepel egy *toggle* címke, ami a trigger aktivizálódásakor kiválasztott állapotba állítja az attribútumként megadott *case* ágat, a vele egy szinten lévőket pedig nem-kiválasztott állapotba teszi.



12. ábra: a leírás fül

13. ábra: a címkék fül

14. ábra: a borító fül

Természetes a fenti lapfüles rendszer nem így nézne ki CSS nélkül.

```
#lapfülesMenü xf|trigger {
    border-left: black solid 1px;
    border-top: black solid 1px;
    border-right: black solid 1px;
    border-bottom: 0px;
    font-weight: bold;
    font-family: Helvetica, sans-serif;
    font-size: .9em;
    margin-right: 5px;
    padding: 3px;
    /* a gombok sarkainak lekerekítése (csak firefox alatt működik)
*/
    -moz-border-radius: .5em .5em 0em 0em;
    border-radius-topright: .5em;
    border-radius-topleft: .5em;
}
/* a lap keretének megadása */
#lap {
    width:500px;
    border-left: solid black 1px;
    border-top: solid black 1px;
    border-right: solid black 1px;
    border-bottom: solid black 1px;
}
```

Az XForms lehetőséget ad arra, hogy új csomópontot szúrjunk be, vagy egy régit töröljünk az adatmodellből. Továbbá tartalmaz egy olyan eszközt, amivel egy csomópont összes gyermekét kilistázhatjuk. Ezeket az eszközöket az alkalmazásomban arra használom, hogy egy filmhez címkéket adjak hozzá, vagy töröljek.

```
<xf:repeat id="repeat-címkék" bind="címke" >
    <xf:output ref="@szöveg"/>
    <xf:trigger appearance="minimal">
     <xf:label class="törlés">X</xf:label>
     <xf:delete ev:event="DOMActivate" bind="címke"
at="index('repeat-címkék')" />
     </xf:trigger>
</xf:repeat>
<xf:input  ref="instance('címkék-proto')/címke/@szöveg">
    <xf:label>új címke: </xf:label>
    <xf:action ev:event="DOMFocusIn">
       <xf:setvalue ref="instance('címkék-
proto')/címke/@szöveg"></xf:setvalue>
    </xf:action>
</xf:input>
<xf:trigger>
   <xf:label>hozzáad</xf:label>
<xf:insert ev:event="DOMActivate" nodeset="instance('film')/címkék"
at="last()" position="after" origin="instance('címkék-proto')"/>
</xf:trigger>
```

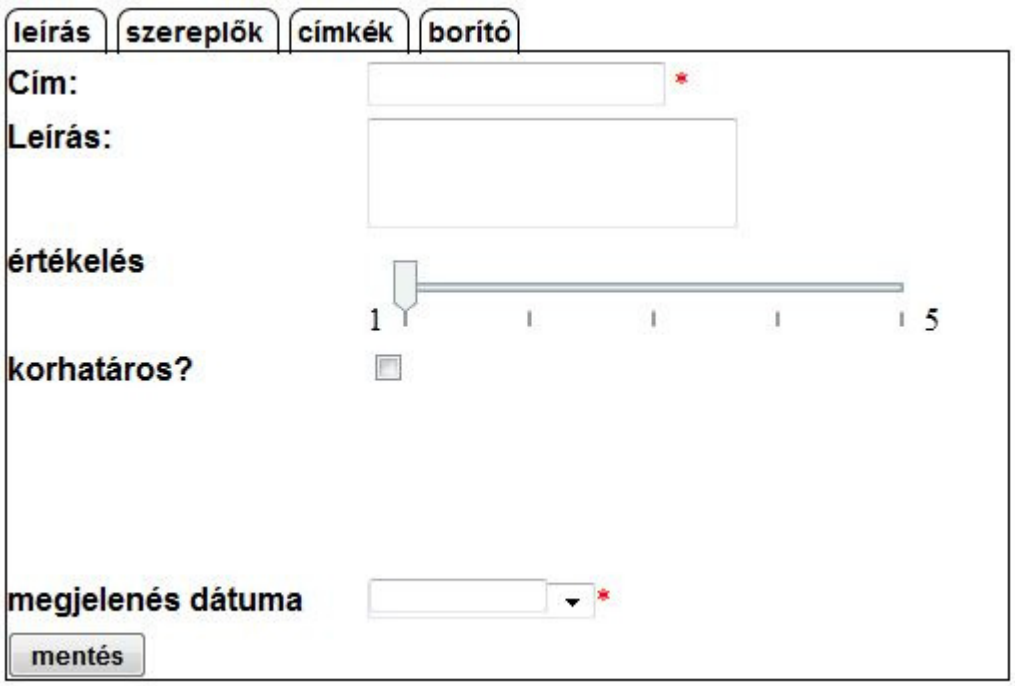
15. ábra: repeat, insert, delete

A *repeat* egyik fontos attribútuma az *id*, amivel indexelni lehet az elemeket. Mindegyik címkéhez hozzárendel egy gombot, ami tartalmazza a *delete* műveletet. Ez a gomb fog gondoskodni arról, hogy az adott címkét kitörölje. Az aktuális címke sorszámát az *index('repeat-címkék')* segítségével kérdezi le. Új címkét beszúrni az *insert* művelet segítségével lehet. Az *at="last()" position="after"* attribútumok azt mondják meg, hogy az új csomópontot a célcsomópont legutolsó gyermeke után szúrja be. Ha a csomópontnak nincs egyetlen gyermeke sem, ami itt annyit jelent, hogy nincs megadva címke, akkor az *origin* attribútum nélkül nem tudna beszúrni csomópontot, mert a beszúrás tulajdonképpen egy másik csomópont lemásolásával történik. Ezért egy külön adatpéldányban létrehoztam, egy egyetlen címkéből álló listát, aminek az értékét egy input vezérlővel módosítok, és ezzel a módosított értékkel szúrom be. Az *origin* tartalma egy sablon csomópont.

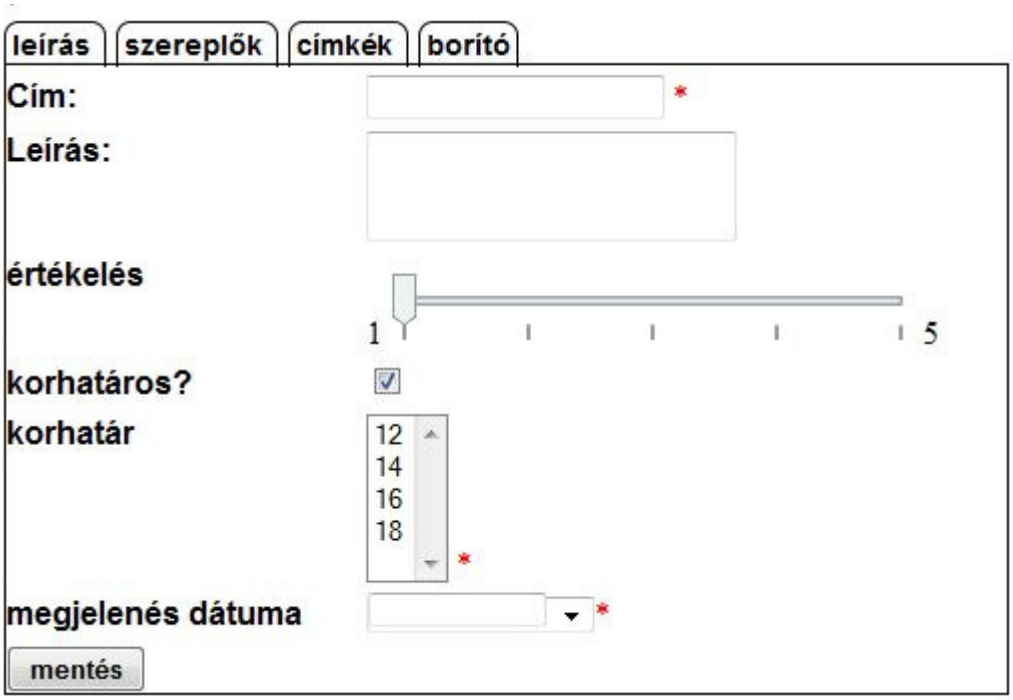
Ezeken kívül lehetőségünk van CSS-el is dinamikus megjelenítésre. Példa rá, hogy a film adatainál lehetőségünk van annak meghatározására, hogy a film korhatáros-e vagy sem. Ha azt adjuk meg, hogy korhatáros, akkor megjelenik egy menü, amiből kiválaszthatjuk az ajánlott életkort.

```
<xf:model>
...
<xf:bind nodeset="/film/korhatár/@korhatáros" type="xs:boolean" >
<xf:bind nodeset=".." required="boolean-from-
string(/film/korhatár/@korhatáros)" relevant="boolean-from-
string(/film/korhatár/@korhatáros)"/>
</xf:bind>
...
</xf:model>
...
<xf:input ref="/film/korhatár/@korhatáros" incremental="true()">
   <xf:label>korhatáros?</xf:label>
</xf:input>
<xf:select1 ref="/film/korhatár" appearance="compact">
   <xf:label>korhatár</xf:label>
   <xf:itemset nodeset="instance('évek')/év">
       <xf:label ref="."></xf:label>
       <xf:value ref="."></xf:value>
   </xf:itemset>
</xf:select1>
```

Az input vezérlő egy logikai értékhez van hozzákapcsolva, ami alapján meghatározódik a korhatár értékének szükségessége. Ha a vezérlőt bekapcsoljuk, akkor a korhatár adat fontos és szükséges lesz (*required* és *relevant*) ellenkező esetben nem.



16. ábra



17. ábra

Ehhez csupán egy egyszerű CSS deklarációt kell megadnunk.

```
*:disabled { visibility: hidden; }
```

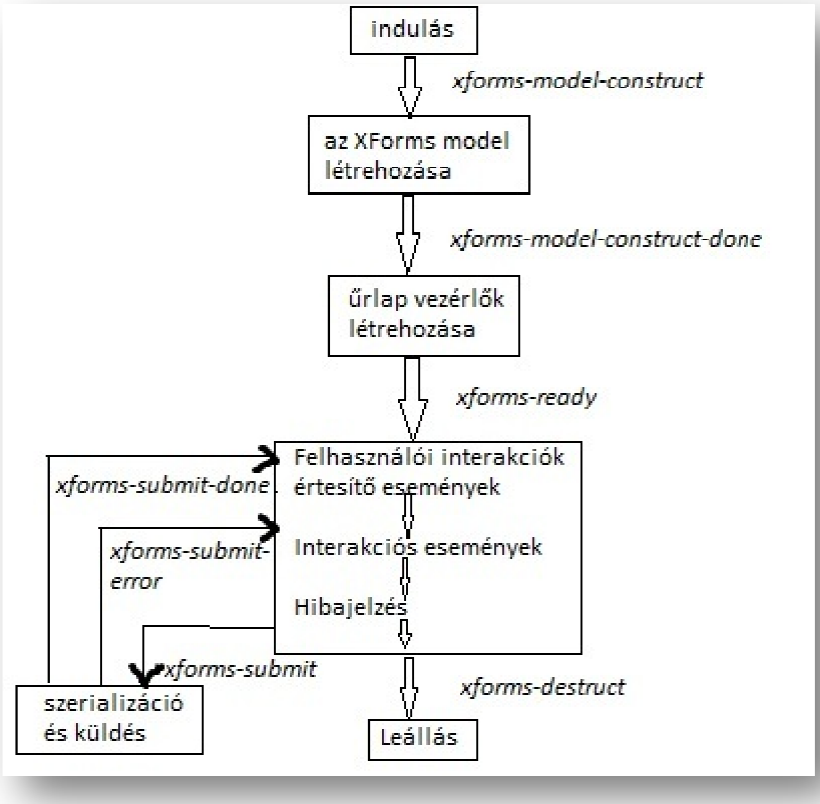
A piros csillag egyes vezérlők után a vezérlőhöz tartozó adatok megadásának szükségességét jelzi. Ezt is CSS-en keresztül adhatjuk meg.

```
*:required::after { content: "*"; color: red; }
```

Műveletek és események

XForms események

Az esemény egy adatszerkezet, ami egy változásról tárol információt. Ilyen esemény, például amikor rákattintunk egy gombra a felhasználói felületen. Ha létrejön egy esemény, akkor az végigmegy az egész dokumentumon, ha közben találkozik egy esemény figyelővel, ami az adott eseményt várja, akkor végrehajtódik az ott megadott utasítás. Ha nincs ilyen eseményfigyelő, akkor az esemény visszakerül a dokumentum gyökeréhez és megszűnik létezni. Az XForms-ban nem csak a felhasználói felület hozhat létre eseményt, hanem maga a modell betöltődése is. Az XForms feldolgozó életciklusa több részre osztható fel.



18. ábra: Az XForms feldolgozó életciklusa

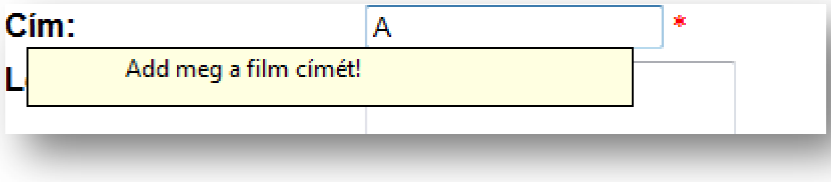
Az ábrán látható, hogy az XForms betöltődése során három esemény is létrejön. Ha adatot küldünk a szerver felé, akkor események segítségével tudunk tájékozódni a küldés állapotáról.

```
<xf:model>
...
<xf:submission id="mentés"
               action="Ment"
               method="post"
               replace="instance">
<xf:toggle case="case-mentés-folyamatban" ev:event="xforms-submit"/>
<xf:toggle case="case-mentés-hiba" ev:event="xforms-submit-error"/>
<xf:toggle case="case-mentés-sikeres" ev:event="xforms-submit-
done"/>
...
</xf:model>
...
<xf:switch>
  <xf:case id="ready"/>
  <xf:case id="case-mentés-folyamatban">
    <p>Várakozás a szerver válaszára</p>
  </xf:case>
  <xf:case id="case-mentés-hiba">
    <p>Hiba a mentés során</p>
  </xf:case>
  <xf:case id="case-mentés-sikeres">
    <p>A mentés sikerült</p>
  </xf:case>
</xf:switch>
```

A *submission* elemben elhelyeztem 3 figyelőt, amik a küldés eseményeire reagálnak, úgy hogy egy switch-case szerkezet ágait manipulálják. A switch-case szerkezet tulajdonképpen a küldés sikerességre vonatkozó információ megjelenítésére szolgál. Amikor rákattint a felhasználó a *sumbission*-höz tartozó gombra, akkor az adatpéldány sikeres validációja után létrejön egy *xforms-submit* nevű esemény, ami a küldésre vonatkozó címkén belül összetalálkozik a megfelelő figyelővel, és az bekapcsolja a *case-mentés-folyamatban* nevű *case* ágat. Ha az adatpéldány érvénytelen, vagy olyan nem kitöltött mezőt tartalmaz, aminek értékét kötelező megadni, vagy a szerver érvénytelen adatot ad vissza, akkor az *xforms-submit-error* esemény jön létre. Ha pedig a modell is rendben van és a válasz is megfelelő akkor az *xforms-submit-done* esemény jön létre.

A felhasználói interakciós események közül a leggyakrabban használt a *DOMActivate* esemény. Ezt aktivizálhatja egy egérkattintás vagy az enter billentyű leütése. Használható triggerekben vagy hiperhivatkozásokban is.

Az értesítő események közé tartoznak az *xforms-help* és az *xforms-hint* események, amelyekkel egy speciális üzenetet rendelhetünk a vezérlőkhöz. Az *xforms-hint* esemény akkor aktivizálódik, amikor az egeret egy vezérlő fölé visszük. Ekkor megjelenik a vezérlő *hint* címkéjében megadott szöveg.



19. ábra: *hint* esemény eredménye

Hasznos események még az *xforms-valid* és az *xforms-invalid* események, amelyek egy érték érvényességének megváltozása után aktivizálódnak. Az *xforms-valid* akkor aktivizálódik, amikor egy érvénytelen érték érvényessé változik, az *xforms-invalid* pedig az ellenkező esetben.

XForms műveletek

Az XForms lehetőséget biztosít különböző műveletek végrehajtására adott esemény bekövetkezésekor. Ezek a műveletek a következők:

message – ez a művelet egy üzenetet küld a felhasználónak, elég tolakodó módon. Az üzenet megakadályozza a további műveleteket, amíg a felhasználó el nem távolítja azt a képernyőről.

```
                                                    xf:message>
```

20. ábra: egy üzenet

setvalue – képes megváltoztatni egy csomópont attribútumának értékét vagy egy csomópont szövegét. Az érték, amit kicserél, lehet a *value* attribútumban vagy lehet a címke tartalma is,

mint szöveg literál. Ha sehol nem adtunk meg értéket, akkor a kiválasztott elem értéke egy nullahosszúságú sztring lesz. A következő kód kitörli az input vezérlő értékét, ha rákattintunk.

```
<xf:input  ref="instance('címkék-proto')/címke/@szöveg">
 <xf:label>új címke: </xf:label>
 <xf:action ev:event="DOMFocusIn">
  <xf:setvalue ref="instance('címkék-
proto')/címke/@szöveg"></xf:setvalue>
 </xf:action>
</input>
```

reset – a *model* attribútumában megadott modell példányait kezdőállapotba állítja.

```
<reset model="film" ev:event="DOMActivate"/>
```

load – egy link által hivatkozott oldalt tölt be vagy egy új ablakba vagy az aktuális oldal helyére. Az URL megadható a *resource* attribútumban vagy egy összekapcsoló attribútum által hivatkozott adatpéldányban.

```
<load resource="http://www.w3c.org" ev:event="DOMActivate"
show="replace"/>
<load ref="linkek/kovetkezoOldal" ev:event="DOMFocusOut"
show="new"/>
```

Egy korábbi példában, amikor címkéket rendelek egy filmhez, a címkék érvényességének ellenőrzését a küldés előtt végzi el a feldolgozó motor. Sokkal szebb megoldást kapunk, ha már a címke hozzáadását sem engedjük meg, ha nem megfelelő formátumú. Egy címkének legalább 3, legfeljebb 15 karakter hosszúnak kell lennie. Ennek ellenőrzésére létrehozok egy új adattípust:

```
<xs:simpleType name="címkeTípus">
  <xs:restriction  base="xs:string">
  <xs:minLength value="3"/>
  <xs:maxLength value="15"/>
  </xs:restriction>
</xs:simpleType>
```

Az új címkét, amit be fogunk szúrni, a *címkék-proto* adatpéldány tartalmazza. A címkéhez megadtam egy attribútumot, ami egy logikai értéket tartalmaz arra vonatkozóan, hogy a címke érvényes-e vagy sem. Ezt az értéket két *setvalue* művelet fogja módosítani, attól függően, hogy az *xforms-valid* vagy az *xforms-invalid* esemény következik-e be. A *trigger action* művelete aktiválódik a gombra való rákattintáskor, de csak akkor, ha az *if* attribútumában szereplő kifejezés igaz értéket ad vissza.

```
<xf:model>
...
  <xf:instance id="címkék-proto">
    <címkék xmlns="">
      <címke szöveg="" érvényes="false"/>
    </címkék>
  </xf:instance>
...
</xf:model>
...
<xf:input ref="instance('címkék-proto')/címke/@szöveg">
  <xf:label>új címke: </xf:label>
  <xf:setvalue ev:event="xforms-invalid" ref="instance('címkék-
proto')/címke/@érvényes" value="'false'"/>
<xf:setvalue ev:event="xforms-valid" ref="instance('címkék-
proto')/címke/@érvényes" value="'true'"/>
</xf:input>

<xf:trigger>
  <xf:label>hozzáad</xf:label>
  <xf:action ev:event="DOMActivate" if="boolean-from-
string(instance('címkék-proto')/címke/@érvényes)">
<xf:insert  nodeset="instance('film')/címkék" at="last()"
position="after" origin="instance('címkék-proto')"/>
</xf:action>
</xf:trigger>
```

Webszolgáltatások

Az XForms egyik nagy erőssége, hogy a kliens képes az adatokat XML dokumentumban küldeni, és fogadni. Ez a képesség kivételesen hasznos a webszolgáltatások területén, ahol szintén XML üzenetek váltódnak.

A következő példákban szeretném bemutatni azt, hogy hogyan lehet használni az XForms-t webszolgáltatásokhoz és azt is, hogy hogyan cserélhető le az adatpéldány a dokumentum többi részének megváltoztatása nélkül.

A példákban http://tv.animare.hu/webservice/tvguide.asmx webszolgáltatást használom, aminek a segítségével lekérdezhető egy csatorna aktuális műsora. Az alkalmazás két lépésből fog állni. Először lekérdezzük a csatornák listáját, majd ebből a listából kiválasztva egyet megkapjuk az aktuális műsort.

A csatorna lista lekérdezéséhez a webszolgáltatás az alábbi XML dokumentumot várja:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ChannelList xmlns="http://tv.animare.hu/webservice/" />
  </soap:Body>
</soap:Envelope>
```

Egy csatornához tartozó műsort pedig az alábbi XML dokumentummal kérhetjük le:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <CurrentProgramme xmlns="http://tv.animare.hu/webservice/">
      <ChannelID>azonosító</ChannelID>
    </CurrentProgramme>
  </soap:Body>
</soap:Envelope>
```

Ennek megfelelően könnyen létrehozható a két adatpéldány, ami a fenti két XML dokumentumot fogja előállítani. Ezeken kívül meg kell adnunk további két példányt az eredmény tárolására.

```
<xf:model id="model-műsor">
  <xf:instance id="CsatornaListaKérés">
    <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Body>
        <ChannelList xmlns="http://tv.animare.hu/webservice/" />
      </soap:Body>
    </soap:Envelope>
  </xf:instance>

<xf:instance id="kiválasztottCsatorna" xmlns="" >
  <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
      <CurrentProgramme xmlns="http://tv.animare.hu/webservice/">
         <ChannelID></ChannelID>
      </CurrentProgramme>
    </soap:Body>
  </soap:Envelope>
 </xf:instance>

<xf:instance id="CsatornaLista" xmlns="">
</xf:instance>
<xf:instance id="program" xmlns="">
</xf:instance>
</xf:model>
```

Az adatpéldány kicseréléséhez egy *submission* elemet kell megadnunk a modellben, amit ha meghívunk, akkor a webszolgáltatásnak elküldi a *CsatornaListaKérés* nevű adatpéldányt és a válaszul kapott XML dokumentumot beleteszi a *CsatornaLista* nevű példányba. A *replace=instance* attribútum mondja meg azt, hogy példánycserét fog végrehajtani.

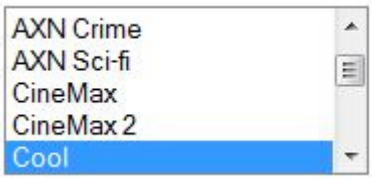
```
<xf:submission id="csatornák"
               method="post"
               action=http://tv.animare.hu/webservice/tvguide.asmx
               ref="instance('CsatornaListaKérés')"
               replace="instance"
               instance="CsatornaLista"
               mediatype="text/xml"
               >
</xf:submission>
```

A webszolgáltatás által visszaküldött dokumentum egy részlete:

```
<soap:Envelope>
 <soap:Body>
  <ChannelListResponse>
   <ChannelListResult>
    <diffgr:diffgram>
     <AnimareTVGuide>
      <ChannelTable diffgr:id="ChannelTable1" msdata:rowOrder="0">
       <ChannelGroupTitle>Magyar</ChannelGroupTitle>
       <ChannelID>97</ChannelID>
       <ChannelGroupID>1</ChannelGroupID>
       <Title>Autonómia Televízió</Title>
       <UrlWebSite>http://www.dunatv.hu/</UrlWebSite>
       <Url>http://tv.animare.hu/default.aspx?channel=97</Url>
      </ChannelTable>
      <ChannelTable diffgr:id="ChannelTable2" msdata:rowOrder="1">
        ...
      </ChannelTable>
      ...
     </AnimareTVGuide>
    </diffgr:diffgram>
   </ChannelListResult>
  </ChannelListResponse>
 </soap:Body>
</soap:Envelope>
```

Látható, hogy a csatornák tulajdonságai a *ChannelTable* elemek tartalmazzák. Ezek közül az azonosítóra (*ChannelID*) és a csatorna nevére (*Title*) van szükségünk. A csatornák megjelenítésére egy lehetőség a *select1* vezérlő használata.

```
<xf:select1 model="model-műsor"
ref="instance('kiválasztottCsatorna')/soap:Body/*/*"
appearance="compact">
    <xf:itemset model="model-műsor"
nodeset="instance('CsatornaLista')//ChannelTable">
        <xf:label ref="Title"/>
        <xf:value ref="ChannelID"/>
    </xf:itemset>
    <xf:send ev:event="xforms-value-changed"
submission="MostJátszott" />
</xf:select1>
```



21. ábra: A csatornák select1-es megjelenítése

Ahhoz, hogy az oldal betöltődésekor egyből megjelenjen a lista, egy műveletet kell beszúrnunk a modellbe.

```
<xf:send submission="csatornák" ev:event="xforms-ready"/>
```

Amikor egy elemet kiválasztunk a select1 vezérlőn belül akkor automatikusan meghívódik a *MostJátszott* azonosítóval rendelkező *submission,* ami lecseréli a *program* nevű példány tartalmát, a kiválasztott csatornának megfelelően.

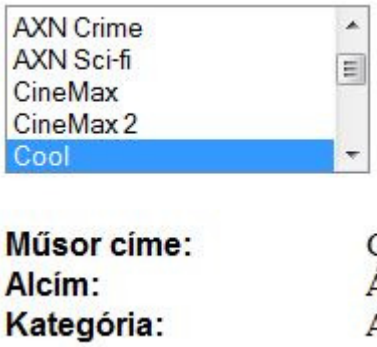
```
<xf:submission id="MostJátszott"
               method="post"
               action="http://tv.animare.hu/webservice/tvguide.asmx"
               ref="instance('kiválasztottCsatorna')"
               replace="instance"
               instance="program"
               mediatype="text/xml"/>
```

A *program* nevű adatpéldány tartalmazza a kiválasztott csatornával kapcsolatos adatokat. A *Table1* nevű elemek tartalmazzák a műsorokkal kapcsolatos információkat. Az első ilyen elem az aktuális műsort tartalmazz, a többi pedig az azt követőket.

```
<soap:Envelope>
 <soap:Body>
  <CurrentProgrammeResponse>
   <CurrentProgrammeResult>
    <diffgr:diffgram>
     <AnimareTVGuide>
      <Table1 diffgr:id="Table12" msdata:rowOrder="1">
       <StartDateTime>2010-05-12T17:00:00+02:00</StartDateTime>
       <Title>Zorro</Title>
       <SubTitle>Aki kapja, marja!</SubTitle>
       <Category>Amerikai tévéfilmsorozat</Category>
       <Url>
http://tv.animare.hu/tvmusor.aspx?id=a587c87636287140429504e9c6a308b
d13
       </Url>
      </Table1>
      <Table1 diffgr:id="Table12" msdata:rowOrder="1">
        ...
      </Table1>
      ...
     </AnimareTVGuide>
    </diffgr:diffgram>
   </CurrentProgrammeResult>
  </CurrentProgrammeResponse>
 </soap:Body>
</soap:Envelope>
```

Ezeket az adatokat *output* vezérlő segítségével egyszerűen kilehet íratni:

```
<xf:output model="model-műsor" value="instance('program')//Title">
    <xf:label>Műsor címe:</xf:label>
</xf:output>
    <br/>
<xf:output model="model-műsor"
value="instance('program')//SubTitle">
    <xf:label>Alcím: </xf:label>
</xf:output>
    <br/>
<xf:output model="model-műsor"
value="instance('program')//Category">
    <xf:label>Kategória: </xf:label>
</xf:output><br/>
```



22. ábra: az aktuális műsor.

Látható, hogy az XForms nagyon jól együttműködik a webszolgáltatásokkal az XML-nek köszönhetően. Az alkalmazást egyszerűen és gyorsan elkészíthetjük. Az adatpéldányok cserélgetésének lehetősége miatt nem kell az oldalt állandóan újratölteni, ha egy másik csatorna műsorát akarjuk megtekinteni.

Összefoglalás

Az XForms egy egyszerű könnyen elsajátítható nyelv, amely jelentősen csökkentheti scriptek írásának szükségességét. Láthattuk azt is, hogy milyen jól elválasztja az adatokat a megtelítéstől, így sokkal áttekinthetőbbé teszi a kódot. Mivel képes az egész oldal újratöltése nélkül az űrlap adatainak cserélgetésére a szerverrel, és képes az adatok érvényességét ellenőrizni, továbbá egyszerűbb számításokat elvégezni, jelentősen csökkentheti a szerver terheltségét. Az XForms-nak számos előnye van az elavult HTML 4.01-es űrlapokkal szemben. De a legnagyobb hátránya, hogy egyik böngészőben sincs natív módon implementálva, és valószínűleg nem is lesz. Erre megoldást az XHTML2 megjelenése hozhat, amely már tartalmazni fogja az XForms szabványt. A probléma csak az vele, hogy a W3C előbb az HTML5-öt akarja elkészíteni, ezért az XHTML2 fejlesztését leállította. Ezért az XForms elterjedésére még sokat kell várnunk. Addig használhatjuk a böngésző kiterjesztéseket, vagy szerver oldali átalakítókat.

Köszönetnyilvánítás

Szeretnék köszönetet mondani a szakdolgozat létrejöttéhez nyújtott segítségért témavezetőmnek, dr. Adamkó Attilának.

Irodalomjegyzék

Könyv

Micah Dubinko: *XForms Essentials*. O'Reilly Media, 2003.

T. V. Raman: *XForms: XML Powered Web Forms.* Addison-Wesley Professional, 2003

Vég Csaba: *Instant Java/Java EE/SOA*. Logos 200 Bt., 2007

Internet

XForms 1.1 W3C ajánlás
http://www.w3.org/TR/xforms11/

HTML űrlapok - az alapok
http://webszabvany.blog.hu/2009/04/10/html_urlapok_az_alapok

Űrlapok HTML dokumentumokban
http://www.w3.org/TR/html401/interact/forms.html

XForms adat fogadása Java-ban
http://www.ibm.com/developerworks/xml/library/x-xformstipjava/

A webes szabványok modellje — HTML, CSS és JavaScript
http://webszabvany.blog.hu/2008/07/17/a_webes_szabvanyok_modellje_html_css_es_javascript

IBM Workplace Forms ***2.7*** információs központ
http://publib.boulder.ibm.com/infocenter/wf/v2r7m0/index.jsp