Egyszerű weboldalak készítése DocBook XML és CSS használatával

Hogyan építsünk egyszerű tartalomszolgáltató weboldalakat DocBook XML és CSS használatával?

Eredetileg arra találták ki a világhálót, hogy információinkat könnyen elérhetővé tudjuk tenni. Manapság a webfejlesztők a stílusra és a marketingre koncentrálnak, mégis ugyanúgy jelen van az igény a könnyen és gyorsan összeállítható weboldalak létrehozására, mint amikor ***Tim Berners-Lee*** először megálmodta a ***HTML*** szabványt. Az ő szemléletét vettem át a ***DocBook XML*** és a ***CSS*** használatával, illetve kerestem néhány könnyen elérhető linuxos eszközt, melyek lehetővé teszik, hogy egyszerű, tartalomra koncentráló weboldalakat hozzak létre – a „szegény ember tartalomkezelő rendszerét".
Beágyazott szoftvereket fejlesztek. A ***HTML***, az ***XML***, a ***CSS*** és maga a web általában nem lényeges a munkámhoz. Nem vagyok olyan bensőséges viszonyban a ***HTML*** szabvány furcsa kiszámíthatatlanságával, mint a processzorokkal, hálózati csatolókártyákkal, soros portokkal. Mégis, ma mindenbe beszivárog a világháló. Egy beágyazott processzor ***Linux*** alatti futási képességeiről gyakran úgy győződünk meg, hogy rákeresünk az interneten.
Én az ügyfeleimet, s ők engem a világhálón keresnek meg. Bár a ***JavaScript***, a böngészőfüggetlen ***HTML***, a ***CSS***, ***PHP***, ***Ruby on Rails*** stb. szaktudás nem túl lényeges, mégis, egy bizonyos szintű HTML ismeret (illetve néhány alapvető eszköz használatának képessége, amivel egyszerű, de hasznos weboldalakat lehet gyártani) egyre inkább elengedhetetlen a szoftverfejlesztéshez csak úgy, mint sok más munkához. A ***DocBook XML*** eszközt ad a kezünkbe, amivel a tartalomra koncentráló dokumentációt tudunk készíteni, oly módon, hogy ezt sokféleképpen, például weboldalakon is könnyedén fel tudjuk használni.

Ez a megközelítés számos összetevőből áll, melyek nem teljesen függetlenek egymástól. Még ha nem is fogadható el általánosságban a szemléletmódom, egyes részei könnyen kiemelhetőek és használhatóak más összefüggésben is.
Magamat szoftveres eszközöket használó embernek tartom. Valószínűleg van számos webes fejlesztőkörnyezet, amely mindent megcsinál, ha megismeri valaki a használatát. Bizonyára van erre számos ***Eclipse*** bővítmény is. A hatékony célszerszámoknak azonban általában meglehetősen meredek tanulási görbéje van, ami csak akkor kifizetődő, ha sokat dolgozik valaki az adott eszközzel.
Ez a cikk nem a ***DocBook XML***-ről szól, hanem arról, hogy hogyan építsünk egyszerű módon weboldalakat, melyek ***DocBook XML*** szerkezetéhez a megjelenítést ***CSS*** adja meg. Nem vagyok webfejlesztő, és olyan eszközök megtanulására vállalkozom csak, melyek széles körben elterjedtek. Azaz: ***vim*** a szerkesztőprogramom, ***m4*** vagy ***Perl*** a makrófeldolgozóm és ***HTML tidy*** segítségével ellenőrzöm a helyes szintaxist – azaz ugyanazokkal az eszközökkel írom a szoftvert, mint amivel a dokumentációt. Az elmúlt néhány év folyamán az alap ***XML***-t, különösen is a ***DocBook XML***-t egyre inkább az alapvető ismereteim közé soroltam be.
Egy egyszerű ***DocBook XML*** cikk szövegsablont mindig készenlétben tartok a ***vim***ben, hogy egyből elő tudjam venni, ha indíttatást érzek valami olyan technikai szöveg megírására, ami hosszabb egy e-mailnél. A ***DocBook XML*** sablon használatával a tartalomra, a lényegre tudok koncentrálni. Világos és kifejező dokumentumokat tudok készíteni anélkül, hogy különösebb erőfeszítésbe kerülne a megjelenítés.
Nemrégiben fedeztem fel, hogy némi ***CSS*** segítséggel a ***DocBook XML*** dokumentumok közvetlenül megtekinthetőek bármely ***CSS***-képes böngészővel – mindenféle ***HTML***-lé alakítás nélkül, ami nagyban megkönnyíti a weboldalamon történő publikálást. Bonyolultabb szövegek esetén pl. az ***OpenOffice.org***-ból is lehet ***DocBook XML***

kimenetet előállítani. Egyre több alkalmazás képes manipulálni és előállítani ***DocBook XML*** formátumot, például az ***OpenOffice.org*** közvetlenül tudja ezt olvasni, sőt tetszőleges elterjedt formátumba át is alakítja, mint például ***HTML***, ***PDF***, ***Microsoft Word*** stb. Az ***XML*** egyik fő célja (melyet meglehetősen nehéz lenne tetten érni az egymással versengő ***XML*** alapú szövegszerkesztő-formátumokban) a tartalom és megjelenés elkülönítése. Szívemből tudom támogatni ezt az alapelvet.
A weboldalakon meg szoktam különböztetni navigációs és tartalmat mutató részeket. Ezeket szándékosan, fájl szinten is elkülönítem egymástól. A tartalomra koncentráló weboldalak legtöbbször nem igényelnek a navigációt, különálló dokumentumként működnek. Ma már ezeket ***DocBook XML***-ben írom meg. Régebben a ***HTML*** szabványt használtam, de akkor is törekedtem arra, hogy a tartalom és a navigáció ne keveredjen. Első lépésben egy ***HTML*** alapú megjelenítő és navigációs keretrendszert hozok létre. Elkészítem a weboldal ***HTML*** indexfájlját, amelyben ***HTML*** kereteket (***frame***) használok a három részegység: a fejléc, a menü és a fő rész elkülönítéséhez. A keretek a legtöbb webfejlesztő szemében nem számítanak szalonképes megoldásnak, talán azért sem, mert könnyen elérhető velük olyan megjelenítés, hogy a más által elkészített honlaptartalmat valaki a sajátjaként tálalja. A navigációt is gátolja a használatuk, és a fogyatékkal élő embertársaink számára is kevéssé barátságosak. Mindazonáltal én nem tudok ehhez mérhető weblap-összeállító módszerről, amellyel a navigáció és a megjelenítés ilyen egyszerűen, frappánsan elkülöníthető lenne. Vannak eszközök, melyekkel hasonló hatások érhetőek el, de az általam ismertek mindegyike beolvasztja a navigációs és a megjelenítési elemeket a „tartalom” kategóriájába. Az a célom, hogy a honlap tartalmi részeit ***DocBook XML*** struktúrában tudjam fejleszteni, melyet csak néhány stíluslap erejéig módosítok. Fontos számomra, hogy valahol máshol különüljön el egymástól a megjelenítés és a navigáció.
Van egy másik eretnek mellékhatása is ennek a megközelítésnek: nevezetesen az, hogy egyáltalán nincs szüksége webszerverre. Mindezt egyetlen böngésző segítségével is fel lehet építeni, ki lehet próbálni – nem kell webszervert telepíteni. Amikor készen vagyunk, csak kiírjuk az egészet egy ***CD***-re, és bárhol megtekinthető az eredmény egyetlen webböngészővel.
Az index oldalam lényegében ennyi:

```
<frameset class="frame" cols="140,*" bordercolor=
➥"#000000"
frameborder="0" framespacing="0">
 <frame class="frame" src="margin.html"
 ➥name="Margin" scrolling="no"
marginwidth="0" marginheight="0"
 <frameset class="frame" rows="100,*"
 ➥bordercolor="#000000"
frameborder="0" framespacing="0">
   <frame class="frame" src="header.html"
   ➥name="Header" scrolling="no"
marginwidth="0" marginheight="0" />
   <frame class="frame" src="home/index.xml"
   ➥name="Body" scrolling="auto"
marginwidth="0" marginheight="0" frameborder=
➥"0" />
 </frameset>
</frameset>
```

Ez három részt határoz meg a böngésző felületén. A menü van baloldalt, a fejléc fent és a tartalmat hordozó fő rész a fennmaradó (jobb alsó) területen. Fontos szerepe lesz a továbbiakban a `name` által meghatározott címkéknek (`margin`, `header`, `body`), hiszen ezzel hivatkozhatunk majd rájuk.
A fejlécem meglehetősen visszafogott, alapvetően csak ennyi:

```
<body class="header" id="body-header">
 <div class="header">
  <h1 class="header">My Title</h1>
 </div>
</body>
```

A `class` és az `id` kulcsszavak teszik majd lehetővé a stílusok alkalmazását a ***CSS*** révén.
A menüsáv számára létrehozott margó is hasonlóan egyszerű:

```
<body class="margin" id="body-margin">
 <div class="menu-box">
  <div class="menu" id="home">
    <a href="home/index.xml"
    ➥target="Body">Home</a>
  </div>
...
 </div>
</body>
```

A `class` és az `id` kulcsszavak megint csak a ***CSS*** stíluslapok alkalmazhatóságát szolgálják. Minden menüpontot körülveszünk egy `menu-box` stílusú téglalappal – ezeket tetszőleges számban ismételhetjük egymás után. Ennek stílusát egyéni ízlésünknek megfelelően bármikor átalakíthatjuk a ***CSS*** segítségével. A kódrészlet szerint a hivatkozás tartalma a menüpontokban megadott célpontban (`target`), azaz a “***body***” címke által meghatározott keretben (a keretrendszer fő részében) jelenik meg, lecserélve az ott korábban megjelenített tartalmat.
Az alábbi ***CSS*** kódot használom a kivilágítható menügombok előállítására:

```
div.menu-box {
 display: block;
 border-width: 2pt;
 border-color: color_bkgr !important;
 border-style: inset ;
}
div.menu {
 border-style: inset ;
 border-width: 5px ;
 background: color_menu_bkgr1 !important;
 border-color: color_menu_bkgr !important;
 color: color_bkgr !important ;
 font-weight: bold;
```

```
 font-size: 8pt;
 height: 14pt ;
 Width: 110pt;
 vertical-align: middle;
 x-margin: 5pt;
 x-padding: 5pt;
 text-align: center;
 padding-left: 5pt;
}
div.menu:hover {
 position: relative;
 top: 1px;
 left: 1px;
 border-color: color_menu_bkgr1;
 background-color: color_menu_bkgr;
}
a.menu { text-decoration: none }
```

Ezek tehát a nem-tartalmi rész kulcsfontosságú összetevői. A menüsávok hierarchikusan egymásba ágyazhatóak. Egyegy menüpont céljának (`target`) `margin`-ra állítása azt eredményezi, hogy a hivatkozott tartalom (praktikusan egy másik menürendszer) az oldalsó margó sávjában, azaz az eredeti menürendszer helyén jelenik meg. Ez a menüváltás akárhányszor megismételhető. A Microsoft Internet Explorer *CSS* kezelése (különösen is a pozicionálás) sajnos esetenként hibás, így a használatakor némi megjelenítésbeli eltérés észlelhető a megfelelően működő böngészőkhöz képest. Böngészőfüggetlen pozicionálás ugyan megoldható, de szörnyen bonyolult – az is nehezíti, hogy az *MSIE 7*-es verziója úgy old meg több *CSS* kérdést, hogy a legtöbb korábbi probléma-megkerülés használhatatlanná válik.
A háttérszínekkel kapcsolatban is óvatosságra intenék mindenkit. Életem egy szakaszát azzal töltöttem, hogy próbáltam kitalálni, miként lehet kiküszöbölni azt a kis fehér csíkot, ami a menüsáv és a fő rész között keletkezik a Microsoft Internet Exploreren, amikor háttérszínt állítok be – de hiába. Ez a cikk nem arról szól, hogy hogyan váljunk szakértőkké a különleges böngészőfüggetlen webfejlesztés terén, hanem egy egyszerű utat ajánl a tartalom szép megjelenítéséhez, függetlenül attól, hogy milyen böngészőben nézik. A pixelről pixelre megegyező, böngészőfüggetlen *CSS* megoldások számos böngésző számára igen nagy kihívást jelentenek.
Eddig nem szóltam a *HTML* fejlécről, sem olyan apróságokról, mint például az, hogy a ***color_menu_bkgr*** nem *HTML*/*CSS* szín, holott a kódban úgy szerepel, mintha az lenne.
HTML oldalaink, mint az ***index.html***, a ***header.html*** és a ***margin.html*** érvényes *HTML* fejlécet igényelnek, amiben egy vagy több hivatkozás is szerepel a kívánt *CSS* stíluslapra, például ennek megfelelően:

```
<link rel="stylesheet" type="text/css"
➥href="/css/stylesheet.css"
title="default">
```

A korábban idézett *CSS* részlet éppen ebben a ***stylesheet.css***-ben található. Ez a kód hívhat még egyéb *CSS* fájlokat is, amikre pl. azért lehet szükség, hogy egyes alapértelmezett *DocBook CSS* értéket felülírjunk vagy kiegészítsünk. Számos *CSS* stíluslap elérhető a *DocBook XML* számára – ezek közül néhány a *DocBook Wiki* oldalakon szerepel. Magam a ***badgers-in-foil***-t használom (lásd a cikk végén, a hivatkozások közt). A ***badgers-in-foil*** stíluslap tette lehetővé számomra, hogy különösebb erőfeszítés nélkül jelenítsek meg *DocBook XML* cikkeket különböző böngészőkben.
Minden *XML* oldalon két stíluslap-hivatkozást kellett beillesztenem az *XML* fejlécbe:

```
<?xml-stylesheet href="/css/docbook-
➥css/driver.css" type="text/css"?>
<?xml-stylesheet href="/css/stylesheet.css"
➥type="text/css"?>
```

A második sor nem feltétlenül kötelező, de erősen ajánlott: ennek révén lehetséges az eredeti *DocBook XML* stíluslapok alapértelmezett értékeinek kibővítése vagy felülírása anélkül, hogy megváltoztatnánk az eredeti stíluslap-fájlt.
A keretrendszer, az *XML*, a *HTML* körítések és számos ismétlődő összetevő előállítására ***m4*** makrófeldolgozót használok. A feladat hasonló könnyedséggel elvégezhető lenne *Perl* vagy ***bash***/***sed*** segítségével is. Ez teszi lehetővé számomra standard fejlécek, színek és más szövegrészek ügyes helyettesítését ***m4*** makrók segítségével. Például a `color_bkgr` is egy ***m4*** makró, amely lehetővé teszi az ugyanilyen nevű szövegrészek lecserélését az általam választott háttérszínre a honlaprendszer összes oldalán.
Ugyanazt a keretrendszert használom fel újra és újra, amikor egy-egy új honlap elkészítésére kell vállalkoznom. Ha egy új honlapot eltérő tartalommal, címekkel, színekkel szeretnék elkészíteni, akkor néhány változtatást kell eszközölnöm a makrókon. Ezek aztán addig bonyolódtak, míg végül azon kezdtem gondolkozni, hogy az előfeldolgozást illetően átálljak *Perl* használatára az ***m4*** helyett. Remek lehetőségnek tartom a feldolgozást követően a *HTML* ***tidy*** általi ellenőrzést, mivel az *XML* és a *HTML* kód előállítását automatizáltan végzem, és ez rejthet buktatókat.
Először is telepítenünk kell a *HTML* ***tidy*** és ***m4*** programokat. Alapvetően ***Debiannal*** és leszármazottaival dolgozom, így a telepítés számomra ennyiből áll:

```
apt-get install tidy
apt-get install m4
```

A legtöbb disztribúció előre csomagoltan tartalmazza az ***m4***-et és a *HTML* ***tidy***-t, de ha forrásból szeretné valaki lefordítani, akkor e programok honlapja megtalálható a cikk hivatkozásai közt.
Haladjunk tovább: egy ***pages.list*** nevű szövegfájlból indulok ki, amely az egyes oldalak alapneveit hordozza a megfelelő típussal együtt, ami lehet *CSS*, *HTML* és *XML*.

```
stylesheet,css
index,html
header,html
margin,html
home,xml
...
```

Az ***m4*** és ***HTML tidy*** futtatására egy rövid héjprogramot használok. Ez minden oldalt végignéz, és – hacsak nem fut hibára – előállítja a kívánt eredményt a megfelelő fájlokba:

```
#!/bin/sh
# $Id:
# $URL:
#dest=../test
dest=..
lname=pages.list
dopage() {
 echo "$1"
 if [ "$2x" == "xmlx" ]; then
  if ! [ -d $dest/$1 ]; then
   mkdir $dest/$1
  fi
  m4 -D_xml $1.m4 | tidy -i -xml >$dest/$1/
  ➥index.xml
 elif [ "$2x" == "htmlx" ]; then
  m4 $1.m4 | tidy -i  >$dest/$1.html
 elif [ "$2x" == "cssx" ]; then
  m4 -D_css $1.m4 >/var/www/share/css/$1.css
 else
  echo "Whoops $1 $2"
 fi
}
if [ -f $lname ]; then
 list=`cat $lname | grep -v '#' | awk '{print
 ➥$1}' | tr '\n' ' '`
 for argv in $list ; do
  page=""; fmt=""
  page=`echo $argv | awk -F "," '{print $1}'`
  fmt=`echo $argv | awk -F "," '{print $2}'`
  dopage ${page} ${fmt}
 done
fi
```

Ezzel az ***m4*** révén a standard fejlécek, a stíluslapokra mutató hivatkozások, a makróhelyettesítések és a színek helyettesítése stb. megoldódott, sőt még maguk a menüpontok is automatikusan állíthatóak elő a megfelelő makrókkal.
A fejléc részért felelős ***header.m4*** fájl így fog tehát kinézni:

```
define(_page,header)dnl
include(defs.m4)dnl
include(hdr.m4)dnl
<div class="header">
 <h1 class="header">_title</h1>
</div>
include(ftr.m4)dnl
```

Mint már szó volt róla, nem szükséges webszerver ahhoz, hogy ezeket a keretrendszereket vagy az előállított tartalmak bármelyikét meg tudjuk nézni. A weboldalak megjelenítését azonban többnyire webszerver végzi. Ennek beállításaiban tulajdonképpen semmit sem muszáj megváltoztatni; mégis, ha az alábbi ***CSS***-beállító sorokat beillesztjük a ***/etc/apache2/conf.d*** megfelelő fájljába, akkor ezzel létrehozunk egy megosztott CSS könyvtárat. Ennek révén több weblap is tudja majd használni ugyanazokat a stíluslapokat, függetlenül a weblapok elérési útjától a webszerver fájlrendszerében.

```
Alias /css /var/www/share/css/
<Location /css>
 Order allow,deny
 Allow from all
 Options Indexes FollowSymLinks MultiViews
</Location>
```

Amit itt láttunk, az egy szoftveres eszközökre építő megközelítés. Ha valaki csak néhány weboldalt készít el minimális tartalommal, nyilván nincs értelme a ***HTML***, ***XML*** fájlok vagy a fejlécek, láblécek automatikus előállításával bajlódni. Ha viszont megnő a szolgáltatott tartalom mennyisége, a változtatások gyakorisága, vagy sok különböző honlapot kell létrehozni, akkor már meghálálja magát az automatizálásra szánt idő.
Igazából alig érintettem a ***DocBook XML***-t. A szövegfeldolgozást már középiskolás éveim alatt elkezdtem egy H8-as gépen, olyan formázóprogramok használatával, mint a ***runoff***, ***nroff*** és ***text***. A tartalom és megjelenés elkülönülése számomra nem más, mint magától értetődő visszatérés nem-***ALAKHŰ*** (***nem-WYSIWYG***) gyökereimhez. (***A betűszó fantáziadúsan magyarított jelentése: Azt Látod, Amit Kapsz, HŰen.***)
Az ***XML*** alapú dokumentumok ***ALAKHŰ*** feldolgozását támogató eszközök is léteznek. Ha kényelmesebbnek tűnik az ***ALAKHŰ*** szövegszerkesztő, akkor a legkézenfekvőbb ***OpenOffice.org***-ot használni, mely ***DocBook XML*** formátumban is tud menteni. Azonban az ***OpenOffice.org DocBook XML*** képességei sem korlátlanok. Amikor egy szépen megformált ***OpenOffice.org*** vagy ***Microsoft Word*** formátumú fájlt ***DocBook XML*** struktúrájúvá alakítunk, a megjelenés néhány jellemzője megváltozhat. A normál ***DocBook XML*** szabvány sokkal inkább a tartalmat és a struktúrát veszi célba, semmint a megjelenítés részleteit. Az ***OpenOffice.org*** nem társít stíluslapot az elmentett ***DocBook XML*** dokumentumhoz, így néhány stíluselem (mint pl. a betűtípus, a betűméret, a beljebb kezdés mértéke stb.) csak a felhasználó által használt ***DocBook XML CSS*** alapján helyettesítődik be. Ha ez nem megfelelő, akkor módosítható vagy felülbírálható a stíluslap egy új, „sorba kapcsolt stíluslappal” (innen az angol név: ***Cascading Style Sheet***, CSS), melyben a kívánt értékek vannak megadva.
Ahogy korábban említettem, a magam részéről teljesen elégedett vagyok a ***badgers-in-foil*** stíluslappal. Ehhez alig kellett hozzányúlnom. Inkább az a célom, hogy minél fájdalommentesebben készíthessem el jól olvasható dokumentumaimat, majd ezek átkerüljenek egy honlapra vagy más, igény szerinti formátumba. Arról is szó esett már, hogy többnyire egyszerű ***DocBook XML*** cikk sablonból szoktam kiindulni, melybe a szövegtartalmat egyszerűn ***Vim*** segítségével írom be. Ez a sablon a ***DocBook XML***-nek csak igazán minimális részét alkalmazza. Néhány ***XML*** alapelvtől eltekintve (mint pl. a kezdő és záró kulcsszavak illeszkedése) az általam beírt szöveg sem tartalmaz a triviális formázásokon túl szinte semmit.

Gyakorlott *DocBook XML* használók az *XML* struktúrák egész arzenálját vethetik be, de a kezdők is egyre kifinomultabb írásműveket készíthetnek, apránként megismerve a használható kulcsszavakat. Megítélésem szerint a *DocBook XML*-t sokkal könnyebb használni, mint a *HTML* szabványt. Mivel az *XML* mereven ragaszkodik a kezdő és záró kulcsszavak egyeztetéséhez és a beágyazás szabályaihoz, így alig lehet váratlan következetlenségbe botlani. A struktúra és a megfelelő szövegfelépítés az *XML* dolga (felsorolások, táblázatok, bekezdések, fejezetek, szakaszok stb. jelölésével). A megjelenítésre és látványra vonatkozó információk már a stíluslapokból derülnek ki. Hozzáértő *CSS* fejlesztők kezei közt az egyszerű *DocBook XML* cikkek elegáns öltözetet nyernek. A célom azonban mégsem elegáns dokumentumok és weboldalak készítése, hanem a szövegtartalmak megjelenítése kifejező és olvasható módon, könnyen és gyorsan, különböző formátumokban. A *DocBook XML* egyre növekvő népszerűségnek örvend a webes dokumentum-formátumok között. Számos nyílt forrású projekt, mint például maga a *Linux kernel* is egyre inkább a *DocBook XML*-re támaszkodik, hiszen ez egy standard dokumentációs forma. A *Linux Dokumentációs Projekt* ajánl egy szerzői útmutatást és egy minta cikksablont, amit gyakran használok magam is, csak úgy, mint más online *DocBook XML* forrásból beszerezhető sablont. *Eric Raymond „DocBook mítosztalanító HOGYAN"-ja („DocBook Demystification HOWTO")* kitűnő magyarázatot ad arra, miért fontos a *DocBook XML*, és miért alkalmas arra, hogy a legtöbb nyílt forrású dokumentációs formátum helyébe lépjen. *Michael Smith* könyve, a *„Fogadd meg a tanácsom: Ne tanuld meg az XML-t"* (*„Take My Advice: Don't Learn XML"*) ehhez hasonlóan értékes: elmagyarázza, hogy a *DocBook XML* érdemben való használatához nem kell *XML*-profivá válni, sem a témához közel álló *XML* technológiák özönét mélységében megismerni. *Norman Walsh* és *Leonard Muellner "Alapvető Kalauz"* (*„The Definitive Guide"*) című műve sokkal több mindent elmond, mint amire valaha is szükségünk lesz, valamint beszél néhány fontos mozzanatról olyan esetekben, amikor kifinomult *DocBook XML* használatba kezdünk bonyolódni. Végül remélem, hogy ez a cikk is világossá teszi, hogy a *DocBook XML* hatékony használata nem bonyolult, és csak minimális mennyiségű új ismeret megtanulását igényli.

Linux Journal 2006., 151. szám

Dave Lynch szoftvertanácsadó. A webfejlesztés, az XML, a CSS és a HTML csak érintőleges kapcsolatban vannak a beágyazott- és rendszerszoftverekkel, melyeket többnyire Linux alá ír, lényegében hiábavaló megélhetési próbálkozásként. Egy másik életében építész – jelenleg épp saját házán dolgozik, amikor épp nem weboldalainak romba döntésével vagy ügyfelei programjainak írásával van elfoglalva.

A CIKK FORRÁSAI

➲ www.linuxjournal.com/article/9263