Mérési jegyzőkönyv – Adatbázisok Laboratórium

V. mérés: XML

|  |  |
| --- | --- |
| Név: | **Szabó Bence Farkas** |
| Neptun kód: | **RF57V5** |
| Feladat kódja: | **30 – VASUT** |
| Mérésvezető neve: | Csapó Tamás |
| Mérés időpontja: | **2018-04-23 12:15** |
| Mérés helyszíne: | **HSZK N** |
| Megoldott feladatok: | **1,2,3,4** |
| Elérhető pontszám (plusz pontok nélkül): | **28p** |

# Mérési feladatok megoldása

## feladat

Az első feladatban egy stíluslapot kellett készíteni, mely kiírja az összes állomás adatait (név, város, átlagos utasszám, kapacitás sztrájk esetén) egy html táblázatba.

### Magyarázat

A feladat megoldásához a list.xsl file tartalmát kellett módosítani. Itt a mintaprogram alapul véve, átalakítva a templateket juthatunk el a megoldáshoz.

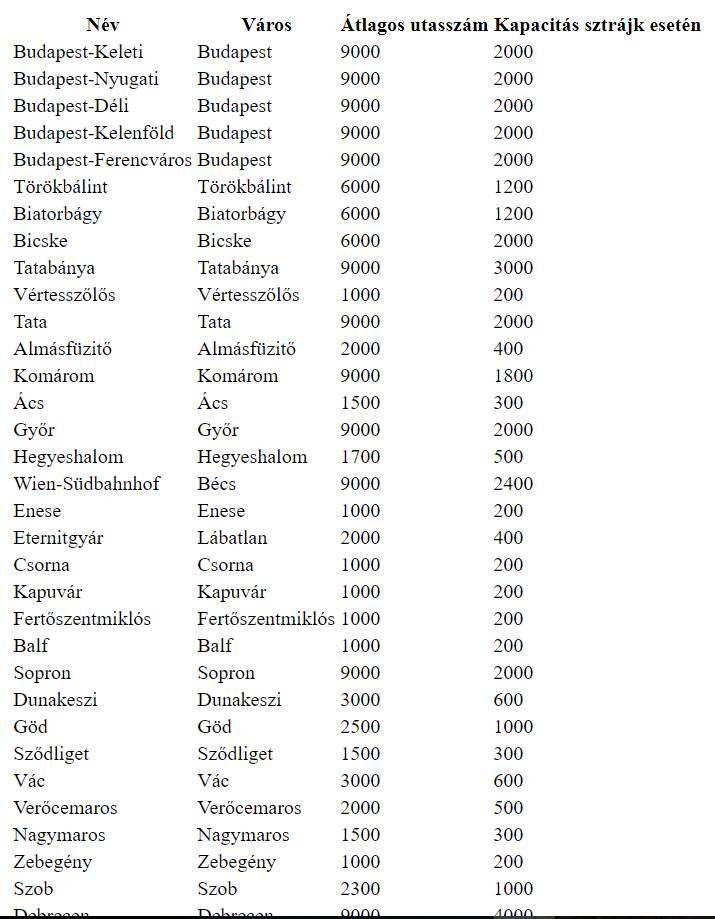
Az első templatet, mely a **„/”** elemre (gyökér elemre) illeszkedik, csak az apply-template utasításnál kellett módosítani, hogy a „**vasut/allomas”** elemekre illeszkedjen.

A második template-ben, mely az „**allomas**” elemre illeszkedik, létrehozzuk a táblázatot, annak fejlécét, majd a feltöltéshez tovább megyünk a gyerek csomópontokra.

A harmadik template-ben az állomás „record” elemein megyünk végig, ahol a feladatnak megfelelően kiválasztjuk a megfelelő attribútumokat, melyek feldolgozását a mintaprogram feldolgozú template-jei végzik.

### Tesztelés menete

A feladat teszteléséhez nyissuk meg a böngészőben a <http://rapid.eik.bme.hu/~rf57v5/xml/list.html> url-t. Az eredményen látható, hogy csak a táblázat kerül kiírásra.



## feladat

A második feladatban a filter.xsl stíluslapot kellett megvalósítani, melyben XML formátumban kilistázzuk azokat az állomásokat, melyek Budapesten találhatók (város attribútumuk Budapest).

### Magyarázat

A feladat megoldásához szintén jó alapul szolgált a kiadott mintaprogram.

Ebben az első templatet módosítjuk úgy, hogy a „**vasut**” elemre illeszkedjen. Itt a copy paranccsal lemásoljuk a csomópontot a kimetenre. A forrás formátumának megtartása érdelében a „**vasut**” csomópontnak kell egy „**element-type**” attribútum, amely a „**dataset**” értéket kapja. Ehhez a template-ben az xsl:attribute paranccsal hozzácsatoljuk a „**vasut**” elemhez az „**element-type**” attribútumot, melynek értékét az xsl:value-of paranccsal kapjuk meg. Ezek után folytatjuk a feldolgozást az „**allomas**” csomóponton.

A második template-ben az „**allomas**” csomópontokat dolgozzuk fel. Hasonlóan az előzőhöz, itt is hozzácsatoljuk az „**element-type**” attribútumot a csomóponthoz, majd az xsl:copy‑of paranccsal lemásoljuk azokat a „**record**” típusú attribútumokat, melyek „varos” gyermeke a „Budapest” értéket tartalmazza (másra nincs is szükség, mert a copy-of megtartja az eredeti fastruktúrát a kimeneten).

### Tesztelés menete

Az eredmény ellenőrzéséhez nyissuk meg a [http://rapid.eik.bme.hu/~rf57v5/xml/filter.xml](http://rapid.eik.bme.hu/~rf57v5/xml/filter.xml%20) url-t. A kimeneten láthatjuk a kész XML struktúrát.



## feladat

Ebben a feladatban az index.xsl stíluslapot kellett megvalósítani, mely a list.xsl stíluslapot felhasználva, önálló html dokumentumként jeleníti meg az állomásokat táblázatos formában.

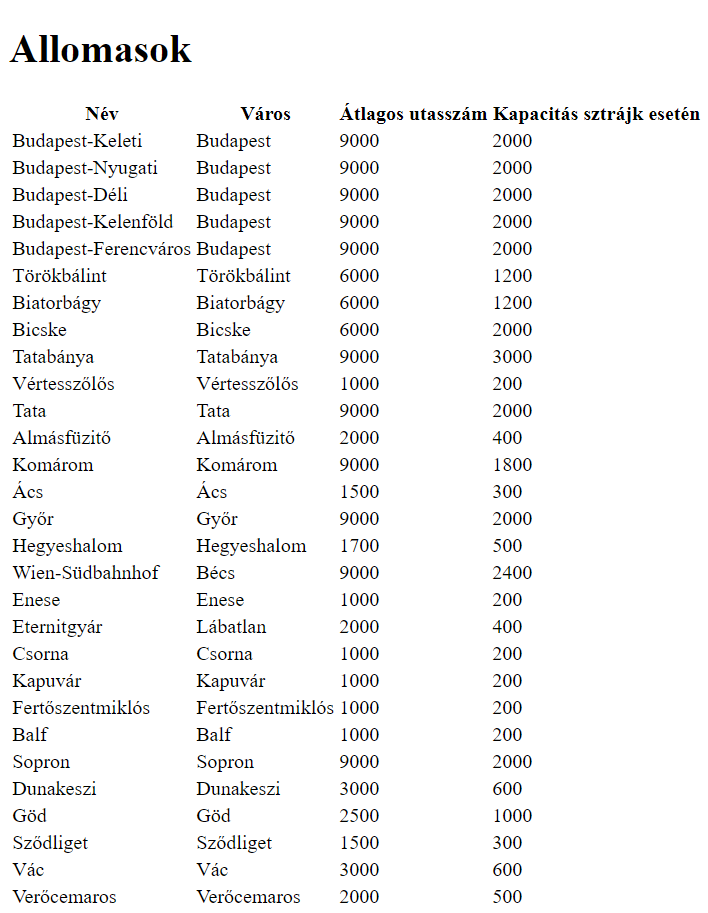
### Magyarázat

A megoldáshoz módosítani kellett az index.xsl-ben található template-et úgy, hogy felüldefiniáljuk a list.xsl root elemre illeszkedő template-jét. Ezt a match="//allomas" attribútum beállításával tehetjük meg.

A következő lépés a táblázat kiírása. Ehhez meg kell hívnunk a list.xsl egyik template-jét. Ezt módosítanunk kell úgy, hogy hivatkozni tudjunk rá (tehát a list.xsl-ben a második template-et, mely az „**allomas**” csomópontokra illeszkedik, ellátjuk egy **name** attribútummal). Ezek után az xsl:call-template paranccsal meghíjuk a list.xsl megfelelő template-jét, amely majd beszúrja a táblázatot.

### Tesztelés menete

A megoldást a <http://rapid.eik.bme.hu/~rf57v5/xml/index.html> url-en érhetjük el. Itt látható, hogy az index.xsl-ben kialakított html oldalba sikerült beszúrni az állomásokat tartalmazó táblázatot:



## feladat

A negyedik feladatban a join.xsl stílusnapot kellett megvalósítani, mely kiírja az állomások listáját és az állomásokhoz tartozó megállások adatait XML formátumban. Az állomás attribútumként tartalmazza saját, mesterséges kulcsát, gyerek csomópontként pedig a név, város, átlagutas mezőket, illetve a vonatok gyerekben szerepelnek az itt megálló vonat adatai melyek a vonatszám, érketés és indulás idejét tartalmazzák attribútumként (ha valamely időpont null, mert az állomás kezdő-, vagy végállomás, azt nem kell kiírni).

### Magyarázat

A megoldáshoz a példaprogram join.xsl stíluslap template-jei módosítására és kiegészítésére volt szükség. Az első template-ben az illeszkedést biztosító match attribútumot kellett a „/vasut” értékre módosítani, elletve az xsl:apply-templates hívást, hogy a "allomas/record" csomópontokon folytassa a feldolgozást.

A második template-ben a mintának megfelelően feltöltjük az állomás adatait. Az állomás id-ját attribútumként, a többi adatot gyerek csomópontként adjuk hozzá. A vonatok lekérdezéséhez létrehozunk egy xsl:variable változót, mely az állomás id-ját tartalmazza. Ez után az xsl:apply-templates utasítással megkeressük a „megall” csomópontok közül azokat, melyeknek „**allomas\_id**” gyerekcsomópontjuk megegyezik az állomásunk id-jával:

<xsl:apply-templates select="/vasut/megall[@element‑type='recordset']/record[allomas\_id=$all\_id]"/>

A harmadik template-ben (módosítva azt, hogy a „**megall/record**” csomópontokra illeszkedjen) kell létrehozni a vonat adatait. Ehhez egy „**vonat**” csomópontot hozunk létre, mely az adatokat attribútumok formájában tartalmazza. Az első attribútum a **vonatszam**, melyet az xsl:value-of parancs segítségével kapunk meg.

Ahhoz, hogy az indulás és érkezés ideje csak akkor jelenjen meg, ha létezik is további template-ekre van szükség. Ezekre nagyon jó példa a list.xsl feldolgozó template-jei, melyek figyelik, hogy az adott csomópont értéke null e. A kiíratáshoz az xsl:apply-templates hívást használva folytatjuk a feldolgozást az „**erk**”, illetve „**ind**” csomópontokon.

Az érkezés és indulás feldolgozása ugyan azon a módon történik:

<xsl:template match="record/erk[@is-null='True']" priority="0.8"></xsl:template>

<xsl:template match="record/erk" priority="0.4">

<xsl:attribute name="erk">

<xsl:value-of select="." />

</xsl:attribute>

</xsl:template>

Itt az első template-ben megnézzük, hogy az adott csomópont null e, ha igen végrehajtjuk a parancsokat (esetünkben nem csinálunk semmit). Ha nem igaz a feltétel (azaz nem null), haladunk tovább a következő template-re, ahol létrehozunk egy attribútumot és értéket adunk neki. A megoldás ugyan az az indulás illetve érkezés feldolgozásánál is, így összesen 4 db új template-et kellett létrehozni.

### Tesztelés menete

A feladat kimenete a <http://rapid.eik.bme.hu/~rf57v5/xml/join.xml> url-en érhető el, ahol látható a felépített XML struktúra:

