

Marek Debnár, Marek Vician

Budovanie digitálnych textových zbierok 1.

(Technologické minimum)

Univerzita Konštantína filozofa v Nitre
2025

doc. Mgr, Marek Debnár, PhD.

Mgr. Marek Vician, PhD.

Vedeckí recenzenti:

doc. Mgr. Richard Změlík, PhD.

Prof. PhDr. Dalimír Hajko, DrSc.

Jazyková redakcia:

Gabriela Sedmáková

ISBN: 978-80-558-2289-1

Táto práca bola podporená Agentúrou na podporu výskumu a vývoja na základe Zmluvy č. APVV-20-0414.



AGENTÚRA
NA PODPORU
VÝSKUMU A VÝVOJA

Obsah

Predslov	3
Digitálny text	5
ASCII	6
Riadiace znaky (0-31 a 127)	7
Tlačiteľné znaky (32-126)	8
Proces (de)kódovania v ASCII	8
Unicode	10
Kódové body (code points)	11
Formy kódovania (code forms)	11
Unicode roviny (Unicode planes)	12
Znaky vs. glyfy	13
Súborové formáty	14
XML	19
Výhody XML	21
Sémantická jasnosť	21
Interoperabilita a znovupoužiteľnosť	22
Strojová čitateľnosť	22
Transparentnosť a uchovávanie	23
Komunita a štandardizácia	23
Špecifikácia XML	24
Základné stavebné prvky XML	24
Štruktúra XML dokumentu	31
Sémantická expresivita XML	35
TEI (Text Encoding Initiative)	39
ELTeC úprava TEI	46
GIT - distribuovaný systém na kontrolu verzií	51
Webové technológie	53
HTML	55
CSS	61
TEI Publisher	64
Technológie	64
Ako TEI Publisher funguje	65
Bibliografia	69

Predslov

Digital humanities alebo digitálne humanitné vedy (ďalej DH) v posledných rokoch priťahujú stále viac pozornosti, a to nielen v zahraničí, kde sú prístupy DH pomerne etablované, čo vidieť napríklad v zavedení samostatných študijných odborov na univerzitách, či existencii DH centier na národných akadémiách vied, ale aj v priestore strednej Európy. Ako každá, relatívne nová, dynamicky sa rozvíjajúca oblasť, aj DH v súčasnosti vykazujú pomerne vysokú mieru sebareflexie. Spreádzajú ich úvahy o metodológii, formulujú vlastné vedecké otázky a transformujú do novej podoby predmety skúmania takmer všetkých humanitných a sociálnych odborov. Digitálne technológie rozšírili nielen metódy, ktoré majú vedci k dispozícii, ale aj druhy otázok, ktoré si môžu klásť – od rozsiahlej analýzy textov a digitálnych edícií, až po mapovanie historických údajov v GIS a šírenie výskumu prostredníctvom webu. Rozmanitosť, ktorú spomíname sťažuje jednotnú a univerzálne platnú definíciu DH (je možné ich nájsť neúrekom), čo je aj dôvod, prečo sa predkladaný text s touto otázkou nezaoberá na meta-úrovni, t.j. na úrovni hľadania všeobecnej definície, ale využitím konkrétneho príkladu DH výskumu v našom prostredí. Týmto príkladom je Digitálna zbierka slovenskej prózy (DISPRO), ktorá vznikla ako výstup projektu Agentúry na podporu výskumu a vývoja č. APVV-20-0414. Projekt bol zameraný na aplikovaný interdisciplinárny výskum v digitálnych humanitných vedách a jeho výstupom je online dostupná a komplexne anotovaná fulltextová digitálna zbierka slovenskej prózy do roku 1955 (s ohľadom na autorské práva a GDPR, t.j. textov, na ktoré sa nevzťahujú autorské práva). Výsledkom však nie je len samostatná digitálna zbierka, ktorá je nevyhnutná pre digitálny výskum literatúry ako takej, ale aj možnosť využiť technológie jej budovania a výstupy na ďalšie individuálne využitie. Zbierka je totiž spracovaná a anotovaná podľa európskych štandardov (ELTeC) tak, aby na ňu bolo možné aplikovať pokročilé metódy počítačového spracovania literárnych textov označované súhrnne pojmom „dištančné čítanie“. Práve tento konkrétny prístup podľa

nás načrtáva cestu a zároveň slúži ako príklad k porozumeniu toho, aké miesto DH zastávajú v súčasnom vedeckom diskurze, konkrétne v digitálnom výskume literatúry.

Práve spomínaná otvorenosť celého projektu DISPRO, ktorá umožňuje užívateľom ďalšie využitie, dopĺňanie a prispôbovanie základnej zbierky textov a technológií nás viedla k tomu, aby sme pre záujemcov pripravili dve učebnice nazvané: Budovanie digitálnych textových zbierok 1. (Technologické minimum), ktorá je vstupom do problematiky a Budovanie digitálnych textových zbierok 2 (Teória a aplikácia), ktorá je aplikáciou poznatkov z prvej časti na konkrétnom textovom materiáli. Prvá časť predstavuje prehľad kľúčových digitálnych nástrojov a infraštruktúr, ktoré formujú súčasný výskum v oblasti humanitných vied a skúma, ako tieto technológie pretvárajú kontúry tejto oblasti. Druhá časť sa zameriava viac na teoretický kontext a konkrétnu aplikáciu týchto nástrojov. Primárnu cieľovú skupinu, pre ktorú sú tieto učebnice, ako aj samotná digitálna zbierka DISPRO (www.dizpro.sk) určené, tvoria literárni vedci, širšia obec humanitných vedcov z príbuzných oblastí, čitateľská obec, katedry slovenského jazyka a literatúry doma a v zahraničí, učitelia a študenti slovenského jazyka.

Digitálny text

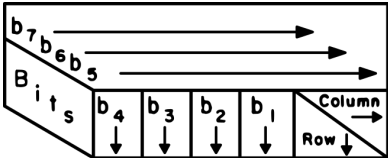
Jadrom mnohých projektov v digitálnych humanitných vedách je digitálny text. Na rozdiel od analógového textu, ktorý je nemenný a statický, v digitálnom texte možno vyhľadávať, analyzovať ho, transformovať a zdieľať s rýchlosťou a presnosťou. Tento posun umožňuje výskumníkom aplikovať výpočtové metódy, automatizovať rozsiahlu textovú analýzu a pristupovať k rozsiahlym korpusom, s ktorými by bolo prinajmenšom nepraktické narábať v ich analógovej podobe. V tejto kapitole sa venujeme otázkam ako, čo znamená, že text je „digitálny“, aké sú formáty, normy a postupy, ktoré umožňujú ich tvorbu, kódovanie a ďalšie vedecké využívanie.

V prvom rade by sme sa mali bližšie vysvetliť, čo myslíme samotným pojmom digitálny. Z technického hľadiska je digitálny text tvorený informáciami reprezentovanými diskretnými jednotkami - zvyčajne ako binárne číslice (bity), ktoré kódujú údaje pomocou iba dvoch stavov: 0 a 1. Na rozdiel od analógových reprezentácií, ktoré sa menia kontinuálne a môžu zachytiť jemné stupňovanie významu alebo signálu (napríklad drážky vinylovej platne alebo tieňovanie kresby uhlíkom), digitálne reprezentácie rozdeľujú informácie na kvantifikovateľné časti, čo umožňuje ich ukladanie, prenos a spracovanie s vysokou presnosťou a reprodukovateľnosťou.

Táto diskretizácia je základom digitálnych technológií. Či už ide o obraz, zvukový záznam alebo text, informácia sa musí najprv zakódovať do digitálnej podoby - štruktúrovanej, počítačmi interpretovateľnej, postupnosti bitov. Tento proces vždy zahŕňa rozhodnutia o formáte, štandardoch kódovania a štrukturálnych konvenciách.

Ako teda reprezentovať znaky nejakého textu - písmená, číslice, interpunkčné znamienka, symboly, atď. - ako binárne údaje, ktoré môže počítač ukladať a manipulovať s nimi? Keďže počítače pracujú s číselnými hodnotami, každému znaku v kódovanom texte musí byť priradené špecifické číslo, ktoré ho jednoznačne identifikuje v rámci vopred definovaného kódovacieho systému.

Prehistória štandardov kódovania textu siaha do počiatkov dejín výpočtovej techniky, keď hardvérové obmedzenia a nedostatočná interoperabilita viedli k rozšíreniu proprietárnych a strojovo špecifických znakových sád. Každý výrobca často vyvinul vlastný spôsob reprezentácie znakov, prispôbený potrebám konkrétnych strojov alebo aplikácií. Napríklad dierne štítky používané v prvých počítačových systémoch používali vlastné kódovanie a zariadenia od rôznych výrobcov mali často nekompatibilné metódy zobrazovania a ukladania textu. Táto roztrieštenosť spôsobovala, že výmena údajov bola ťažkopádna a náchylná na chyby, najmä vo viacjazyčnom alebo multiplatformovom kontexte. Potreba spoločného, štandardizovaného spôsobu reprezentácie znakov sa stávala tým naliehavejšou, čím viac sa rozširovalo používanie výpočtovej techniky a najmä so vznikom sieťovej komunikácie. Tento vývoj viedol k iniciatívam ako ASCII v 60. rokoch 20. storočia a nakoniec k vzniku Unicode ako komplexného riešenia problému.



Bits					Column	Row							
b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁							
0	0	0	0	0	0	0	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0	1	2	3	4	5	6
0	0	0	0	1	0	0	0	1	2	3	4	5	6
0	0	0	1	0	0	0	0	1	2	3	4	5	6
0	0	0	1	0	1	0	0	1	2	3	4	5	6
0	0	0	1	1	0	0	0	1	2	3	4	5	6
0	0	0	1	1	1	0	0	1	2	3	4	5	6
0	0	1	0	0	0	0	0	1	2	3	4	5	6
0	0	1	0	0	1	0	0	1	2	3	4	5	6
0	0	1	0	1	0	0	0	1	2	3	4	5	6
0	0	1	0	1	1	0	0	1	2	3	4	5	6
0	0	1	1	0	0	0	0	1	2	3	4	5	6
0	0	1	1	0	1	0	0	1	2	3	4	5	6
0	0	1	1	1	0	0	0	1	2	3	4	5	6
0	0	1	1	1	1	0	0	1	2	3	4	5	6
0	1	0	0	0	0	0	0	1	2	3	4	5	6
0	1	0	0	0	1	0	0	1	2	3	4	5	6
0	1	0	0	1	0	0	0	1	2	3	4	5	6
0	1	0	0	1	1	0	0	1	2	3	4	5	6
0	1	0	1	0	0	0	0	1	2	3	4	5	6
0	1	0	1	0	1	0	0	1	2	3	4	5	6
0	1	0	1	1	0	0	0	1	2	3	4	5	6
0	1	0	1	1	1	0	0	1	2	3	4	5	6
0	1	1	0	0	0	0	0	1	2	3	4	5	6
0	1	1	0	0	1	0	0	1	2	3	4	5	6
0	1	1	0	1	0	0	0	1	2	3	4	5	6
0	1	1	0	1	1	0	0	1	2	3	4	5	6
0	1	1	1	0	0	0	0	1	2	3	4	5	6
0	1	1	1	0	1	0	0	1	2	3	4	5	6
0	1	1	1	1	0	0	0	1	2	3	4	5	6
0	1	1	1	1	1	0	0	1	2	3	4	5	6
1	0	0	0	0	0	0	0	1	2	3	4	5	6
1	0	0	0	0	1	0	0	1	2	3	4	5	6
1	0	0	0	1	0	0	0	1	2	3	4	5	6
1	0	0	1	0	0	0	0	1	2	3	4	5	6
1	0	0	1	0	1	0	0	1	2	3	4	5	6
1	0	0	1	1	0	0	0	1	2	3	4	5	6
1	0	0	1	1	1	0	0	1	2	3	4	5	6
1	0	1	0	0	0	0	0	1	2	3	4	5	6
1	0	1	0	0	1	0	0	1	2	3	4	5	6
1	0	1	0	1	0	0	0	1	2	3	4	5	6
1	0	1	0	1	1	0	0	1	2	3	4	5	6
1	0	1	1	0	0	0	0	1	2	3	4	5	6
1	0	1	1	0	1	0	0	1	2	3	4	5	6
1	0	1	1	1	0	0	0	1	2	3	4	5	6
1	0	1	1	1	1	0	0	1	2	3	4	5	6
1	1	0	0	0	0	0	0	1	2	3	4	5	6
1	1	0	0	0	1	0	0	1	2	3	4	5	6
1	1	0	0	1	0	0	0	1	2	3	4	5	6
1	1	0	0	1	1	0	0	1	2	3	4	5	6
1	1	0	1	0	0	0	0	1	2	3	4	5	6
1	1	0	1	0	1	0	0	1	2	3	4	5	6
1	1	0	1	1	0	0	0	1	2	3	4	5	6
1	1	0	1	1	1	0	0	1	2	3	4	5	6
1	1	1	0	0	0	0	0	1	2	3	4	5	6
1	1	1	0	0	1	0	0	1	2	3	4	5	6
1	1	1	0	1	0	0	0	1	2	3	4	5	6
1	1	1	0	1	1	0	0	1	2	3	4	5	6
1	1	1	1	0	0	0	0	1	2	3	4	5	6
1	1	1	1	0	1	0	0	1	2	3	4	5	6
1	1	1	1	1	0	0	0	1	2	3	4	5	6
1	1	1	1	1	1	0	0	1	2	3	4	5	6

Obrázok 1: Pôvodná tabuľka ASCII kódov (1967)

ASCII

Jedným z prvých a najvplyvnejších štandardov v oblasti digitálnej reprezentácie textu je kódovacia schéma ASCII (American Standard Code for Information Interchange). Bola

vyvinutá začiatkom 60. rokov 20. storočia pod záštitou Amerického národného normalizačného inštitútu (ANSI) a jej cieľom bolo poskytnúť konzistentný spôsob reprezentácie textu v počítačoch, komunikačných prístrojoch a iných digitálnych zariadeniach.

Podstatou ASCII je 7-bitové kódovanie znakov. To znamená, že na reprezentáciu každého znaku sa používa sedem bitov, čo umožňuje celkovo 128 jedinečných symbolov ($2^7 = 128$). Kódy sú pritom logicky zoskupené do niekoľkých funkčných blokov reprezentovaných číselnými rozsahmi, pričom základné rozdelenie je medzi riadiacimi a tlačiteľnými znakmi.

Riadiace znaky (0-31 a 127)

Prvou veľkou skupinou sú tzv. riadiace znaky, ktoré sa používajú na ovládanie hardvérových zariadení (ako sú tlačiarne alebo terminály), a nie na zobrazovanie symbolov. Patria medzi ne napríklad:

Table 1: ASCII kódy niektorých riadiacich znakov.

Riadiaci znak	Decimálny ASCII kód	Binárny ASCII kód
NUL (nulový znak)	0	00000000
LF (line feed)	10	00001010
CR (carriage return)	13	00001101
DEL (delete)	127	01101010

a ďalších 29 znakov, ktoré sa používali pri interakcii s prístrojmi bez grafických rozhraní. Umožňovali synchronizovať dátové toky, spravovať vyrovnávacie pamäte alebo formátovať text v prostrediach s veľkými hardvérovými obmedzeniami. Postupom času sa síce mnohé z týchto znakov prestali používať alebo sa zmenila ich pôvodná funkcia, ale ich vplyv je stále možné rozpoznať aj v moderných protokoloch, formátoch súborov a v správaní terminálov. Napríklad znaky “Line Feed” (LF, ASCII 10) a “Carriage Return” (CR, ASCII 13) sa stále rôznym spôsobom používajú pre označenie koncov riadkov v textových súboroch alebo terminálových emulátoroch:

- **Unix/Linux** používa pre ukončenie riadkov LF (`\n`).
- **Windows** používa kombináciu CR+LF (`\r\n`).
- **Mac OS** (pred OS X) používa len CR.

Tlačiteľné znaky (32-126)

Tento rozsah zahŕňa všetky znaky, ktoré bežne vidieť na klávesniciach alebo obrazovkách.

Člení sa ďalej na tieto podskupiny:

- **Medzera:** " " (32)
- **Interpunkcia a symboly** (33-126)
 - ! " # \$ % & ' () * + , - . / (33-47)
 - : ; < = > ? @ (58-64)
 - ‘ [] ^ _ “ (91-96)
 - { | } ~ (123-126)
- **Číslice:** 0-9 (48-57)
- **Písmená anglickej abecedy** (65-122)
 - A-Z (65-90)
 - a-z (97-122)

Každému z uvedenej sady znakov je teda pridelený jedinečný binárny kód, ktorý zabezpečuje, že digitálne systémy môžu spoľahlivo ukladať, prenášať a zobrazovať text na rôznych platformách, ktoré sú schopné rozpoznať danú kódovaciu schému.

Proces (de)kódovania v ASCII

Pre ilustráciu si naznačme, ako sa kóduje a dekóduje krátky text "Ahoj!" v ASCII schéme.

Krok 1: *Nájdenie ASCII kódov prislúchajúcich znakov v reťazci "Ahoj!"*

Table 2: ASCII kódy jednotlivých znakov v reťazci "Ahoj!".

Znak	Decimálny ASCII kód	Binárny ASCII kód
A	65	01000001
h	104	01101000
o	111	01101111
j	106	01101010
!	161	10100001

Krok 2: *Kódovanie textu*

Pri kódovaní textu do ASCII počítač prevedie každý znak na zodpovedajúcu binárnu reprezentáciu. Tieto bity sa potom postupne ukladajú alebo prenášajú.

A → 01000001

h → 01101000

o → 01101111

j → 01101010

! → 10100001

Keď ich skombinujeme, celá kódovaná správa v binárnom tvare bude vyzeráť nasledovne:

01000001 01101000 01101111 01101010 10100001

Krok 3: *Dekódovanie ASCII*

Dekódovanie je jednoducho obrátenie predchádzajúceho procesu kódovania. Vychádzame teda zo sekvencie bytov 01000001 01101000 01101111 01101010 10100001, ku ktorým nájdeme znaky, ktoré im zodpovedajú v ASCII schéme:

01000001 → A

01101000 → h

01101111 → o

01101010 → j

10100001 → !

Kombinácia identifikovaných hodnôt potom predstavuje pôvodnú správu “Ahoj!”.

Takéto procesy kódovania a dekodovania prebiehajú vždy, keď užívateľ zadáva nejaký text prostredníctvom klávesnice do počítača, respektíve, keď počítač zobrazuje pre človeka zrozumiteľný text na obrazovke.¹

¹Kódovanie, ako aj dekodovanie vždy prebieha, napríklad, pri písaní textu v textovom procesore ako je MS Word, keďže užívateľom zadaný text sa okamžite zobrazuje na obrazovke.

Unicode

ASCII štandard predstavoval monumentálny krok vo vývoji strojovo čitateľného textu, keďže poskytol jednotnú schému na reprezentáciu textu v prvých počítačových systémoch. Silná stránka tohto štandardu, teda jeho relatívna jednoduchosť, však bola aj jeho slabinou, keďže malý rozsah kódov umožňoval digitálne reprezentovať len obsah v anglickom jazyku a malú podmnožinu technických symbolov. S globalizáciou výpočtovej techniky sa tak obmedzenia ASCII stávali čoraz zjavnejšími.

Ako sme videli, systém ASCII prideluje každému znaku 7 bitov, čo umožňuje reprezentáciu iba 128 jedinečných symbolov. To bolo síce elegantné riešenie pre základný anglický text a riadiace inštrukcie v ranom hardvéri, ale úplne nedostatočné pre kódovanie celého rozsahu znakov používaných v iných svetových jazykoch. Jazyky ako francúzština, nemčina alebo španielčina mohli čiastočne prispôbiť ASCII použitím rozšírených 8-bitových kódovaní², ale táto spleť kódovacích štandardov spôsobila, že výmena textu medzi rôznymi systémami bola nespoľahlivá a náchylná na chyby. To, čo sa v jednom kódovaní mohlo javiť ako zmysluplný text, mohlo byť v inom úplne nezmyselné. Ešte väčším problémom však bolo, že takýmito ad hoc riešeniami nebolo možné zabezpečiť podporu pre nelatinizované písma, ako sú arabčina, hebrejčina, cyrilika, čínština alebo dévanágarí.

Unicode je riešením tohto problému, keďže ide o štandard, ktorého cieľom je priradiť jedinečné číslo (“kódový bod”) každému znaku v akomkoľvek spisovnom jazyku používanom v minulosti i súčasnosti, bez ohľadu na používanú technologickú platformu. Reprezentácie v ňom využívajú maximálne 32 bitov, vďaka čomu je tento štandard schopný kódovať vyše 1.1 milióna znakov. Efektívne používanie v rôznych systémoch a prostrediach je zabezpečené prostredníctvom niekoľkých schém kódovania, z ktorých najbežnejšie sú UTF-8, UTF-16 a UTF-32. Každá z nich ponúka inú rovnováhu medzi kompaktnosťou, kompatibilitou a zložitosťou spracovania. Okrem moderných abecied zahŕňa Unicode aj širokú škálu symbolov, interpunkčných znamienok, matematických operátorov, emotikonov, historických a starovekých písiem a dokonca aj riadiacich znakov, vďaka čomu ide o komplexné riešenie na globálnu reprezentáciu textu.

Pre lepšie porozumenie tomuto štandardu si objasníme niekoľko niekoľko centrálnych pojmov: *kódový bod*, *forma kódovania* a *Unicode rovina*.

²Pozri “ISO/IEC 8859-1”

Kódové body (code points)

Jadrom Unicode je idea tzv. kódového bodu, čo je jedinečný číselný identifikátor priradený každému znaku v schéme tohto štandardu. Môžeme si ho predstaviť ako univerzálnu „adresu“ znaku v obrovskom adresári symbolov. Každý kódový bod je pritom zvyčajne reprezentovaný v tvare U+XXXX, kde XXXX je hexadecimálne číslo (teda číslo v sústave zo základom 16). Pre ilustráciu si vezmime nasledujúce kódovania:

- U+0041 predstavuje veľké latinské písmeno A.
- U+03C0 je grécke malé písmeno π .
- U+20AC zodpovedá symbolu meny euro (€).
- U+1F600 je emotikon usmievajúcej sa tváre (☺).

Kódové body sú abstraktné identifikátory, teda neurčujú, ako znak vyzerá alebo ako má byť uložený v pamäti. Namiesto toho jednoducho poskytujú štandardný spôsob odkazovania na každý znak nad akoukoľvek platformou.

Formy kódovania (code forms)

Hoci kódové body poskytujú univerzálny referenčný systém, počítače musia tieto znaky ukladať a prenášať ako sekvencie bajtov. V Unicode je to zabezpečené prostredníctvom tzv. foriem kódovania, čo sú mechanizmy na prevod kódových bodov na sekvencie bajtov. Medzi najpoužívanejšie z týchto foriem patria:

1. UTF-8 (*Unicode Transformation Format – 8-bit*)

- Kódovanie s premenlivou dĺžkou, ktoré používa 1 až 4 bajty na jeden znak.
- Kompatibilné s ASCII: prvých 128 kódových bodov (U+0000 až U+007F) je kódovaných pomocou jedného bajtu, ktorý korešponduje s ASCII.
- Ideálne pre dokumenty s textom písaným prevažne latinkou.
- V súčasnosti je to najpoužívanejšie kódovanie na webe a v modernom softvéri.

2. UTF-16

- Na kódovanie znakov sa používajú 2 alebo 4 bajty (vo forme jednej alebo dvoch 16-bitových jednotiek).
- Extenzívne používané v systémoch ako Windows alebo Java.

- Efektívne pre jazyky ako čínština alebo arabčina, v ktorých väčšina znakov leží v základnej viacjazyčnej rovine (*Basic Multilingual Plane* alebo *BMP*³).

3. UTF-32

- Používa pevnú 4-bajtovú sekvenciu pre každý znak.
- Jednoduchá manipulácia pri programovaní, pretože každý znak má rovnakú veľkosť.
- Neefektívne z hľadiska úložného priestoru.

Každá forma kódovania je bezstratová, čo znamená, že pôvodný kódový bod možno vždy obnoviť z jeho zakódovanej postupnosti bajtov. Táto flexibilita umožňuje výber kódovania, ktoré najlepšie vyvažuje efektívnosť ukladania, kompatibilitu a jednoduchosť spracovania.

Unicode roviny (Unicode planes)

Unicode rozdeľuje svoj kódový priestor, ktorý pokrýva vyše milión možných kódových bodov, na 17 rovín, z ktorých každá pozostáva zo 65 536 (2^{16}) kódových bodov:

- **Rovina 0:** Základná viacjazyčná rovina (*BMP*). Obsahuje najpoužívanějšíe znaky vrátane latinky, cyriliky, arabčiny, znakov CJK (čínština, japončina, kórejščina), interpunkčných znamienok a diakritiky.
- **Rovina 1:** Doplnková viacjazyčná rovina (*SMP*). Zahŕňa historické písma, hudobné notácie, emotikony a matematické symboly.
- **Rovina 2:** Doplnková ideografická rovina (*SIP*). Vyhradená pre ďalšie znaky CJK.
- **Roviny 3-13:** Vyhradené na budúce použitie.
- **Rovina 14:** Doplnková rovina pre špeciálne účely (*SSP*). Obsahuje riadiace znaky používané pre výbery variantov.
- **Rovina 15 a 16:** Vyhradené pre oblasti na súkromné použitie (*PUA*), ktoré umožňujú vlastné, neštandardné definície znakov.

Prostredníctvom tohto členenia priestoru znakov Unicode kategorizuje a prideluje zdroje rôznym potrebám písiem a symbolov z rôznych časových období a kultúr.

³Pozri nižšie.

Znaky vs. glyfy

Pri narábaní s digitálnym textom je nevyhnutné rozlišovať medzi znakmi a glyfmi, pretože tento rozdiel je vždy na pozadí toho, ako sa reprezentujú, vykresľujú a manipulujú systémy písma na rôznych platformách.

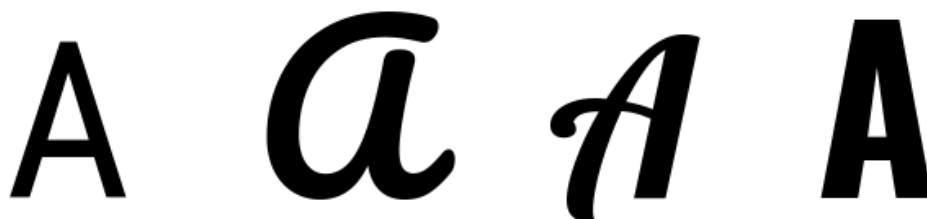
V kontexte Unicode je **znakom** abstraktná, na platforme nezávislá jednotka informácie. Je to symbol, ktorý vyjadruje špecifickú sémantickú alebo fonetickú hodnotu a je jednoznačne identifikovaný kódovým bodom. Dôležité je pritom uvedomiť si, že znaky nie sú viazané na žiadnu konkrétnu vizuálnu formu.

Príklady znakov:

- Veľké písmeno A (U+0041)
- Číslica 5 (U+0035)
- Znak nového riadku (U+000A)
- Grécke písmeno π (U+03C0)
- Emotikon ☺ (U+1F600)

Vzhľadom na určitý systém kódovania, akým je napríklad Unicode, sú znaky vo všetkých technologických prostrediach kódované jednotne a konzistentne, čo z nich robí univerzálne stavebné kamene textového obsahu v digitálnom prostredí.

Na druhej strane **glyf** je vizuálne stvárnienie alebo grafická reprezentácia znaku. Glyfy sú to, čo skutočne vidíte na obrazovke alebo vytlačenej stránke. Vzhľad glyfu pritom závisí od mnohých faktorov ako je použitý font, štylistické variácie (tučné písmo, kurzíva, malé písmená), vykresľovacie prostredie alebo sádzací systém.



Obrázok 2: Rôzne glyfy znaku U+0041 (“A”)

Pochopenie rozdielu medzi znakmi a glyfmi je z praktického hľadiska kľúčové pri kódovaní textu, keďže Unicode je výhradne systém na reprezentáciu znakov, nie glyfov. Podobne je to dôležité pri vyhľadávaní a indexovaní textov, keďže aj tu sa znaky a ich zoskupenia musia identifikovať vždy bez ohľadu na ich vzhľad. A napokon, sémantické znaky, nie ich grafická podoba, sú podstatné pre čítačky obrazoviek, ktoré používajú ľudia s rôznymi formami zrakového postihnutia.

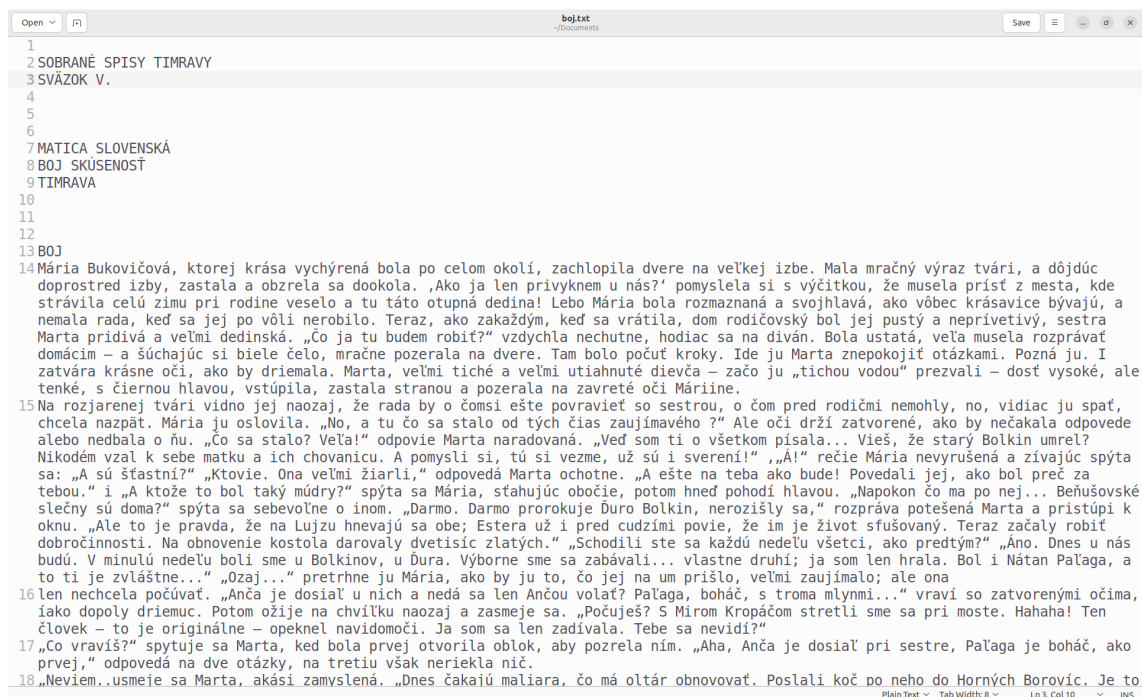
V skratke, systémy kódovania textu sú konvencie, ktoré zavádzajú vzťahy medzi znakmi a ich binárnou reprezentáciou, a sú úplne nezávislé od toho, ako ten-ktorý znak vyzerá na zobrazovacích zariadeniach. Dosadzovanie a vykresľovanie správnych glyfov je úlohou fontov a vykresľovacích systémov, ktoré môžu v rôznych prostrediach prezentovať jeden a ten istý digitálny text rôznym spôsobom.

Súborové formáty

Základnou jednotkou ukladania údajov v digitálnych systémoch je súbor, čo je samostatná kolekcia informácií, ktoré možno ukladať, vyhľadávať a manipulovať s nimi pomocou softvéru a hardvéru. Vo svojej podstate súbor predstavuje postupnosť bajtov, ktoré môžu kódovať čokoľvek od obyčajného textu a obrázkov až po programový kód alebo zložité štruktúrované údaje. Súborné sú pomenované, uložené v adresároch a zvyčajne sú spojené so špecifickými formátmi, čo je štandardizovaná metóda kódovania informácií v počítačovom súbore, ktorá určuje, ako sú údaje štruktúrované, uložené a interpretované softvérom. Definuje pravidlá, ktoré určujú, čo súbor obsahuje a ako majú programy tieto údaje čítať a zapisovať.

Z hľadiska problematiky digitálneho textu môžeme rozlišovať medzi obyčajnými textovými (*plain text*) súbormi, formátmi podporujúcimi značkovanie (*markup* formáty) a binárnymi formátmi. Každý z nich má svoje silné stránky, obmedzenia a dôsledky pre podporu Unicode a dlhodobú prístupnosť.

Textové súbory sa vyznačujú jednoduchosťou a univerzálnosťou. Sú čitateľné pre človeka a možno ich otvoriť a upravovať v akomkoľvek základnom textovom editore bez ohľadu na používaný operačný systém. Ich štruktúra je jednoduchá, čo uľahčuje ich vytváranie, zobrazovanie a spracovanie pomocou širokej škály programovacích nástrojov. Keďže neobahujú zložité formátovanie, sú textové súbory obzvlášť vhodné na ukladanie obsahu, pri



Obrázok 3: Textový súbor zobrazený v textovom editore Gedit

ktorom nie je dôležité rozvrhnutie a vizuálna prezentácia. Tento minimalizmus prispieva aj k ich výnimočnej prenosnosti a robustnosti, pretože sú do veľkej miery odolné voči problémom spôsobeným zmenami softvéru alebo štandardov súborových formátov.

Surový digitálny text zakódovaný v Unicode umožňuje reprezentáciu obsahu v prakticky akomkoľvek ľudskom jazyku, takže je vhodný pre viacjazyčné korpuse. Keďže však v takomto textovom súbore chýbajú vnútorné štrukturálne indikátory, interpretácia sa často spolieha na externé znalosti alebo dodatočné nástroje (napr. regulárne výrazy alebo programovacie knižnice určené na analýzu textu).

Markup formáty vychádzajú z čisto textových formátov tým, že do textového obsahu vkladajú štrukturálne alebo sémantické informácie pomocou formálneho systému značiek alebo anotácií. Ide o formáty, ktoré sú strojovo aj ľudsky čitateľné a ponúkajú premostenie medzi surovým textom a štruktúrovanými údajmi.

Markup formáty sa vyznačujú niekoľkými kľúčovými vlastnosťami. Po prvé, používajú špeciálnu syntax - napríklad značky v ostrých zátvorkách alebo dvojice kľúčov a hodnôt - na anotovanie štruktúry alebo významu obsahu. To umožňuje zachovať logické členenie v texte vrátane prvkov, ako sú kapitoly, odseky, dôraz a hypertextové odkazy. Tieto formáty sú často samopopisné, obsahujú metadáta a deklarácie kódovania priamo v súbore,



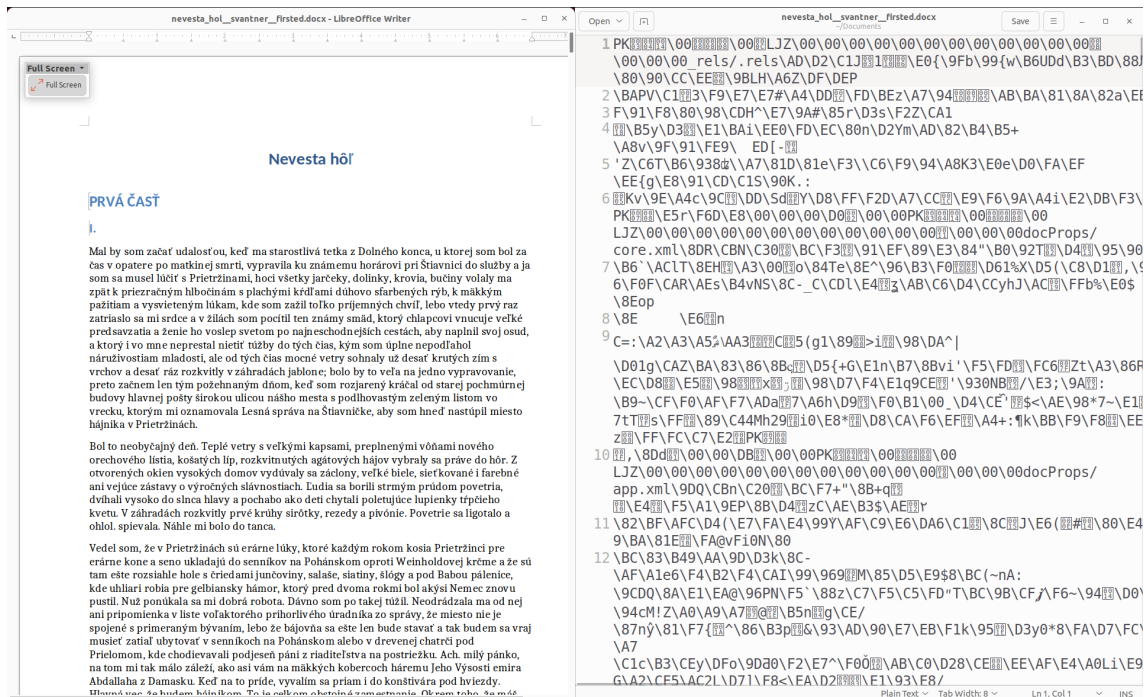
Obrázok 4: XML súbor zobrazený v textovom editore Gedit

čo pomáha pri ich interpretácii a spracovaní. V dôsledku toho sú obzvlášť vhodné na automatizované úlohy, ako je validácia, transformácia a extrakcia údajov.

Markup formáty využívajúce značkovanie sú kľúčové pre mnohé pracovné postupy v oblasti digitálnych humanitných vied, najmä formát TEI XML, ktorý umožňuje kódovať štruktúru a sémantiku textov na ľubovoľne detailnej úrovni.

Napokon **binárne formáty** ukladajú údaje v kompaktných, často neprehľadných sekvenciách bajtov, ktorých čítanie a interpretácia si vyžaduje špecializovaný softvér. Tieto formáty môžu byť optimalizované na výkon alebo funkcie špecifické pre určitú výpočtovú platformu, ale obetujú transparentnosť a dlhodobú použiteľnosť, keďže softvér a operačné systémy sa neustále vyvíjajú.

Formáty binárnych súborov majú vlastnosti, ktoré ich odlišujú od textových alebo markup formátov. Nie sú priamo čitateľné pre človeka; po otvorení v textovom editore sa zvyčajne zobrazia ako spleť nečitateľných znakov. Napriek tomu binárne súbory často obsahujú bohaté vnútorné štruktúry vrátane informácií o rozložení, metadát a v mnohých prípadoch aj vložených zdrojov, ako sú písma alebo obrázky. Tieto formáty sú zvyčajne úzko prepojené s konkrétnymi softvérovými aplikáciami a pri presnej interpretácii a vykresľovaní sa spoliehajú na konkrétne verzie používaného softvéru. Hoci mnohé binárne formáty do



Obrázok 5: Binárny docx súbor zobrazený v špecializovanom programe Libreoffice (naľavo) a v textovom editore Gedit (napravo)

určitej miery podporujú Unicode, implementácia tejto podpory sa môže výrazne líšiť, čo môže viesť k potenciálnym nezrovnalostiam v kódovaní alebo zobrazovaní textu v rôznych prostrediach.

XML

V digitálnych humanitných vedách nie je výber formátu na reprezentáciu textových údajov neutrálnym rozhodnutím, pretože určuje, čo môžeme s textom robiť, ako ho interpretujeme a ako ho zdieľame s ostatnými. Medzi voľby, ktorá sú zrejme najbližšie bežnému užívateľovi, patria formáty textových procesorov, ako napríklad .docx programu Microsoft Word alebo .odt súbory používané v OpenOffice. Tie ponúkajú vizuálne orientované prostredie, v ktorom môžu používatelia písať, upravovať a formátovať texty bez potreby hlbších technických znalostí. Funkcie ako tučné písmo, kurzíva, poznámky pod čiarou a nadpisy sú vďaka intuitívnemu užívateľskému rozhraniu ľahko použiteľné a spoluprácu s ďalšími ľuďmi zjednodušujú integrované funkcie komentovania alebo systémy sledovania zmien. Pri bežnom narábaní s textom v digitálnom prostredí sú vďaka takejto jednoduchosti používania procesory jasnou voľbou.

Táto voľba však so sebou nesie určité obmedzenia, pre ktoré nie sú textové procesory, resp. súborové formáty, ktorými tieto programy reprezentujú texty, vhodné pre ciele, ktoré sledujeme v digitálnych humanitných vedách. Tieto obmedzenia nie sú len technickými prekážkami, ale ovplyvňujú aj spôsob interpretácie, zdieľania a uchovávanía textov vo vedeckej práci, ktorá čoraz viac závisí od štruktúrovaných, pre počítače zrozumiteľných údajov.

Jedným z najzásadnejších problémov je, že textové procesory sú navrhnuté s ohľadom na vizuálnu prezentáciu textov, nie vzhľadom na ich sémantickú zrozumiteľnosť. Formátovacie funkcie, ktoré tieto programy poskytujú, sú zamerané na to, ako text vyzerá pre čitateľa: tučné písmo pre zvýraznenie, kurzíva pre nadpisy, zalomenie riadkov pre odseky. Táto prezentácia však v sebe nenesie žiadne informácie o význame alebo funkcii danej časti textu. Tučným písmom zvýraznený výraz v programe Word môže označovať rečníka v divadelnej hre, postavu v románe alebo nadpis v odbornom článku, čo stroj,

bez ďalšej informácie, nemôže vedieť. Absencia sémantického značenia veľmi sťažuje extrakciu, analýzu alebo opakované spracovanie textu pomocou výpočtovej techniky. Aj keď je vizuálne formátovanie konzistentné, základná štruktúra súboru je zvyčajne neprehľadná, keďže je uložená ako zbierka binárnych súborov, ktoré je ťažké analyzovať bez špecializovaných nástrojov.

Okrem toho sú súbory textových procesorov často nekonzistentné a idiosynkratické. Používatelia volia rôzne formátovanie v závislosti od osobných zvykov, inštitucionálnych šablón alebo predvolených nastavení softvéru. Jedna vedkyňa môže používať kurzívu pre názvy kníh, iný môže používať úvodzovky. Niektorí môžu ručne vkladať zalomenia riadkov, aby vytvorili dojem medzier, iní sa spoliehajú na štýly. Tieto nezrovnalosti sa v spoločných alebo rozsiahlych projektoch rýchlo hromadia, takže automatizované spracovanie alebo analýza sú bez rozsiahleho čistenia a štandardizácie nespoľahlivé.

Ďalšou nevýhodou je netransparentnosť verziovania zmien súborov textového procesora. Word síce ponúka funkcie ako „sledovanie zmien“, tie však nie sú štandardizované ani prenosné medzi rôznymi platformami.

Z hľadiska uchovávaní nie sú formáty textových procesorov veľmi robustné. Keďže sa spoliehajú na proprietárne alebo poloproprietárne technológie, sú náchylné na zastarávanie softvéru alebo zmeny v predvolenom správaní v jeho rôznych verziách. Súbor .docx vytvorený v programe Word 2007 sa nemusí správať rovnako v novších verziách alebo v open-source alternatívach, čo môže viesť k strate údajov, neželaným zmenám vo formátovaní alebo v rozložení textu.

Napokon, pre projekty digitálnych humanitných vied, ktorých cieľom je publikovať texty na webe, prepojiť ich s metadátami alebo zabezpečiť ich plnotextovú vyhľadateľnosť a analyzovateľnosť, sú súbory textového procesora jednoducho nevyhovujúce. Konverzia súborov .docx do vhodne štruktúrovaných formátov si zvyčajne vyžaduje buď množstvo manuálnej práce alebo použitie externých nástrojov, prípadne vlastných skriptov - ani tie nám však nepomôžu, ak nemá pôvodný súbor konzistentnú štruktúru.

Hoci teda formáty textových procesorov vyhovujú potrebám bežného písania a akademického publikovania¹, ich obmedzenia sa naplno prejavujú vo vzťahu k požiadavkám práce v oblasti digitálnych humanitných vied - konkrétne k potrebe modelovať, analyzovať a

¹Predchádzajúce a nasledovné argumenty však poskytujú dôvody v neprospech týchto formátov aj pre tieto použitia.

uchovávať texty bohatým a štruktúrovaným spôsobom. Práve tu ponúka XML (eXtensible Markup Language) robustnú alternatívu. Je to jazyk navrhnutý na reprezentáciu informácií v štruktúrovanom, pre človeka a počítač čitateľnom formáte. Pri jeho návrhu sa kládol dôraz najmä na jednoduchosť, všeobecnosť a použiteľnosť v prostredí internetu² a vyznačuje sa silnou podporou takmer všetkých ľudských jazykov vďaka kompatibilite s Unicode štandardom.³ [Ide o univerzálne znaky určené na podporu celosvetovej výmeny, spracovania a zobrazovania písaných textov rôznych jazykov a technických disciplín moderného sveta.³ Hoci mal jazyk XML pôvodne slúžiť najmä na reprezentáciu dokumentov, v súčasnosti sa extenzívne používa na reprezentáciu ľubovoľných dátových štruktúr,⁴⁵ napríklad tých, ktoré sa vyskytujú vo webových službách.⁶

Výhody XML

Sémantická jasnosť

Jednou z najvýznamnejších výhod jazyka XML je schopnosť sémantického značkovania. Na rozdiel od súborov textových procesorov, ktoré používajú formátovanie predovšetkým na vizuálnu prezentáciu textu, XML umožňuje explicitné definovanie sémantického významu jednotlivých častí textu. Ak chceme, napríklad, v nejakom texte zaznamenať, že určitý reťazec znakov predstavuje meno autora, prostriedkami XML to dosiahneme tak, že danú pasáž uzavrieme v značke `<author>`⁷, ktorá má vopred definovaný význam.⁸ Týmto sa stane rola daného reťazca v dokumente explicitná a jednoznačná.

Zreteľnejšie to možno vidieť pri komplexnejších príkladoch. Historický dokument môže obsahovať vrstvené úrovne citácií, redakčných a autorských poznámok, odkazov alebo marginálií - každý z týchto prvkov možno presne reprezentovať označením pomocou prostriedkov XML. Vďaka tomu tak výskumníci môžu systematicky vyhľadávať prípady konkrétneho hovorcu, sledovať pomenované entity, identifikovať tematické vzory alebo rozlišovať medzi pôvodným textom a redakčnými zásahmi.

²*Extensible Markup Language (XML) 1.0 (Fifth Edition).*

³*Unicode Standard.*

⁴Fennell, "Extremes of XML".

⁵"What Is XML (Extensible Markup Language)?"

⁶"What Is XML (Extensible Markup Language)?"

⁷Technickým detailom XML sa venujeme ďalej v texte.

⁸V tomto kontexte by mohlo ísť o význam "tvorca textu, ktorého je označený reťazec časťou".

XML

XML tak slúži ako nástroj pre formalizované vyjadrenie vedeckej interpretácie. Zviditeľňuje štrukturálne a interpretačné rozhodnutia, ktoré humanisti často nechávajú v ich tradičnej vedeckej produkcii implicitné. To sa obzvlášť dobre zhoduje s cieľmi tvorby kritických edícií a archívnej práce všeobecne, kde je prvoradá vernosť materiálnemu a intelektuálnemu kontextu.

Interoperabilita a znovupoužiteľnosť

Ďalšou kľúčovou výhodou XML je jeho interoperabilita. Keďže je nezávislý od výpočtovej platformy a riadi sa otvorenými štandardmi, súbory tohto formátu možno používať v širokom spektre softvérových prostredí, od databáz a webových aplikácií až po transformačné systémy a nástroje na vizualizáciu údajov.

Súbor vo formáte XML možno napríklad transformovať do HTML formátu určeného na publikovanie na webe, PDF formátu vhodného pre tlač, formátu ePub používaného v elektronických čítačkách alebo dokonca do formátu JSON na integráciu do webových rozhraní v internetovom prostredí. Tieto transformácie sa zvyčajne realizujú pomocou XSLT⁹ alebo¹⁰ iných transformačných “potrubí”¹¹, vďaka čomu môže jeden súbor slúžiť ako zdroj pre generovanie množstva rôznych výstupov bez vynakladania duplicitnej práce.

Okrem toho, keďže sa XML riadi striktnými pravidlami a súbory v tomto formáte môžeme validovať voči vopred definovaným modelom, je ľahké udržiavať texty dobre utvorené a vnútorne konzistentné. Toto zabezpečuje opakovateľnú použiteľnosť a zdieľateľnosť korpusov pozostávajúcich z XML súborov - nielen pôvodnými autormi, ale aj inými výskumníkmi a inštitúciami.

Strojová čitateľnosť

Vďaka hierarchickej a na pravidlách založenej štruktúre XML, poskytujú súbory v tomto formáte ideálny substrát pre dištančné čítanie, štylometriu, sieťovú analýzu, modelovanie

⁹XSLT (Extensible Stylesheet Language Transformations) je jazyk pôvodne navrhnutý na transformáciu dokumentov XML do iných XML dokumentov alebo iných formátov, ako je HTML, obyčajný text alebo formátovacie objekty XSL. Tieto formáty možno následne konvertovať do formátov, ako sú PDF, PostScript a PNG. Podpora transformácie JSON a obyčajného textu bola pridaná v neskorších aktualizáciách špecifikácie XSLT 1.0. (*XSL Transformations (XSLT) Version 2.0 (Second Edition)*, n.d.)

¹⁰*XSL Transformations (XSLT) Version 2.0 (Second Edition)*, n.d.

¹¹Postupnosť automatizovaných krokov alebo procesov, ktoré konvertujú údaje z jedného formátu alebo štruktúry do iného.

tém a ďalšie metódy používané v digitálnych humanitných vedách. Vhodne anotované texty nám napríklad umožňujú ľahko zodpovedať otázky ako koľko ženských postáv hovorí v slovenských románoch z 19. storočia, ako často sa objavujú odkazy na určité miesta alebo ako sa mení štruktúra dialógov v čase. Na tieto typy otázok je takmer nemožné spoľahlivo odpovedať pri použití formátov textového procesora, ktoré nemajú vnútornú štruktúru potrebnú na to, aby boli vhodnými vstupmi pre automatizované spracovanie.

Transparentnosť a uchovávanie

Keďže XML je čisto textový formát¹², vyznačuje sa transparentnosťou a trvácnosťou. Na rozdiel od proprietárnych formátov textových procesorov možno súbory v tomto formáte otvoriť a čítať v akomkoľvek textovom editore, v akomkoľvek operačnom systéme, bez špeciálneho softvéru.

Vďaka tomuto sa zmeny v súboroch XML dajú presne sledovať pomocou systémov na kontroli verzií, ako je napríklad Git¹³, čo je obzvlášť užitočné v kolaboratívnom vedeckom prostredí. Každá úprava, doplnenie alebo oprava sa stáva súčasťou kontrolovateľnej histórie, čo napríklad umožňuje budúcim výskumníkom pochopiť vývoj digitálneho objektu.¹⁴

To, že XML je čisto textový formát, znamená oddelenie obsahu od prezentácie, čo podporuje čistejšie pracovné postupy a znižuje riziko poškodenia údajov v dôsledku problémov s formátovaním. Prezentácia - či už pre web, tlač alebo mobilné zariadenia - sa dá produkovať nezávisle prostredníctvom súborov štýlov a šablón, pričom základné údaje zostanú nedotknuté.

Komunita a štandardizácia

XML v digitálnych humanitných vedách ťaží zo silných komunít, najmä okolo TEI (Text Encoding Initiative), ktorá poskytuje dobre vyvinutý a vyvíjajúci sa štandard pre textovú vedu. TEI ponúka nielen rozsiahly slovník elementov¹⁵ pre širokú škálu textových funkcií,

¹²Pozri predchádzajúcu kapitolu.

¹³„Git“.

¹⁴Samozrejým benefitom je schopnosť obnovenia predchádzajúcich verzií textov.

¹⁵XML schéme TEI sa venujeme nižšie.

ale poskytuje aj dokumentáciu, príklady, nástroje a komunitu vedcov, editorov a vývojárov, ktorí aktívne podporujú jeho prijatie.

Používaním XML a TEI sa výskumníci zapájajú do ekosystému, ktorý si cení transparentnosť, udržateľnosť a vedeckú dôslednosť. Toto zosúladenie so spoločnými štandardmi zvyšuje hodnotu, udržateľnosť a prístupnosť vlastnej práce, čo uľahčuje jej zdieľanie, uchovávanie a ďalšie rozširovanie.

Špecifikácia XML

Pre efektívne používanie XML formátu je dôležité pochopiť jeho základné princípy: stavebné prvky, z ktorých je vyskladaný každý dokument v tomto formáte a pravidlá určujúce akým spôsobom musia byť tieto prvky usporiadané. Predstavíme si preto oba tieto aspekty XML, pričom sa technickejšie implementačné detaily budeme snažiť prepájať s abstraktnejšími princípmi, ktorými sme v predchádzajúcom texte motivovali adopciu XML pre účely digitálnych humanitných vied.

Základné stavebné prvky XML

Medzi základné zložky akéhokoľvek XML dokumentu patria *elementy*, *atribúty*, *text*, *komentáre*, *procesné inštrukcie*, *CDATA bloky* a *menné priestory*. V tejto časti sa oboznámime s každým z týchto komponentov.

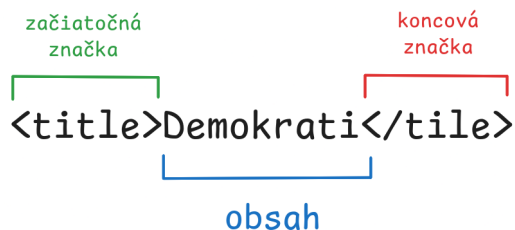
Elementy

Sú základnými jednotkami štruktúry XML, reprezentujú údaje a dávajú im význam prostredníctvom značiek a atribútov. Element sa zvyčajne skladá zo začiatkovej značky¹⁶, obsahu a koncovkej značky¹⁷.

Okrem textu, môžu elementy obsahovať aj ďalšie elementy, atribúty alebo ich rôzne kombinácie:

¹⁶Reťazca znakov ohraničeného ostrými zátvorkami '<' a '>'.

¹⁷Reťazca znakov zľava ohraničeného '</' a zprava ohraničeného '>'.



Obrázok 6: Štruktúra XML elementu

```
<book>
  <title>Dom v stráni</title>
  <author>
    <name>Martin Kukučín</name>
    <dateOfBirth>1860</dateOfBirth>
    <dateOfDeath>1928</dateOfDeath>
  </author>
  <size unit="words">108243</size>
  <pubdate>1912</pubdate>
</book>
```

Prázdné elementy, teda tie ktoré neobsahujú text alebo iné elementy,¹⁸ môžu vystupovať v dvoch ekvivalentných formách:

```
<element></element>
```

```
<element />
```

Mená značiek, ktoré tvoria elementy, podliehajú nasledujúcim obmedzeniam:

- V názvoch sa rozlišujú veľké a malé písmená¹⁹
- Názvy musia začínať písmenom alebo podčiarkovníkom
- Názvy nemôžu začínať reťazcom “xml” (alebo “XML”, alebo “Xml” atď.)
- Názvy nemôžu obsahovať medzery
- Názvy môžu obsahovať písmená, číslice, pomlčky, podčiarkovníky a bodky

Nasledujúci zápis teda nepredstavuje korektný XML element:

```
<title>Dom v stráni</Title>
```

¹⁸Takéto elementy však môžu obsahovať atribúty.

¹⁹To znamená, že <Title>, <title> alebo <TITLE> predstavujú odlišné značky

Atribúty

poskytujú dodatočné informácie o elementoch a umiestňujú sa vo vnútri ich začiatočnej značky, prostredníctvom priradenia `atribút="hodnota"`, pričom hodnoty atribútov sa vždy musia nachádzať v jednoduchých alebo dvojitých úvodzovkách. Prostredníctvom atribútov teda môžeme, napríklad, zaznamenať, že Hana Gregorová je slovenskou autorkou, takto:

```
<author gender="female" nationality="slovak">Hana Gregorová</author>
```

V princípe je možné všetky informácie reprezentovateľné prostredníctvom atribútov kódovať aj prostredníctvom vnorenia elementov, a naopak. Vyššie uvedený element môžeme preformulovať nasledujúcim spôsobom bez akejkoľvek informačnej straty:

```
<author>
  <gender>F</gender>
  <nationality>SK</nationality>
  <name>Hana Gregorová</name>
</author>
```

Atribúty však ponúkajú špecifické výhody v prípadoch, kedy by štrukturálne vnorenie bolo neefektívne alebo sémanticky nevhodné. V prvom rade umožňujú oddeliť dáta od metadát, kde obsah elementov predstavuje samotné dáta a prostredníctvom atribútov reprezentujeme informácie o dátach. Ak by sme chceli napríklad v literárnom texte zaznamenať, že nejaká postava predstavuje protagonistu príbehu, bolo by nevhodné tieto informácie kódovať prostredníctvom samostatných elementov²⁰, keďže by sme tým znemožnili odlíšenie originálneho textu od našich analytických zásahov.

Okrem toho, umožňujú atribúty kompaktnejší a čitateľnejší spôsob reprezentácie jednoduchých informácií - `<character role="protagonist">Šimon</character>` je jednoduchšie a prehľadnejšie, ako alternatíva, pri ktorej by sme použili samostatný element `<role>` pre vyjadrenie toho istého.

²⁰Napríklad ako `<role>protagonist</role>`.

Napokon je použitie atribútov optimálnejšie pre niektoré výpočtové úlohy, ako je filtrovanie (napr. vyhľadávanie všetkých elementov `<character>` s atribútom `role="protagonist"`) alebo validácia dokumentov voči XML schémam.²¹

Text

V XML dokumentoch predstavuje neštruktúrované dáta, ktoré sú obsiahnuté v elementoch. V kontexte digitálnych humanitných vied je to typicky sémantické jadro dokumentu - slová, vety a odseky, ktoré nesú význam. Ide často o pôvodné literárne texty, prepisy, redakčné poznámky a iné, na ktoré s určitým výskumným zámerom uplatňuje vopred definovaný model implementovaný v XML formáte.

Ak si základnú štruktúru XML dokumentu predstavíme ako strom, tak textové údaje sa zvyčajne nachádzajú v jeho listoch. To znamená, že sa vyskytujú v koncových bodoch vetiev stromu, kde nie sú žiadne ďalšie vnorené elementy, čo odráža spôsob, akým XML reprezentuje informácie: vnútorné uzly (elementy) poskytujú štruktúru a klasifikáciu, zatiaľ čo listy (textové uzly) obsahujú skutočný nositeľov analyzovaného významu.

Elementy však môžu mať aj zmiešaný obsah, teda obsahovať tak text ako aj ďalšie elementy. V takom prípade sa text stále považuje za list, ale daný uzol nie je čisto "listový", keďže sa vďaka obsiahnutým elementom ďalej rozvetvuje. Nie každý list XML stromu však musí mať formu textu. Ak by sme napríklad chceli zaznamenať, že na určitom mieste v texte sa v origináli nachádza koniec strany, môžeme na to použiť, napríklad, prázdny element `<pb />`, ktorý by v celkovej štruktúre dokumentu predstavoval listy stromu.

Pre vizuálnu ilustráciu stromovej štruktúry XML dokumentu, si vezmeme nasledujúci fragment úvodu Kukučínovho románu *Dom v stráni*:

V stráni pod Grabovikom, stáby prilepené o strmý bok, stoja domy bratov Bercov. Idúcky z dediny, vlastne mesta, prejdeš najprv popri dome Ivanovom, potom Franičovom a tretí dom je, v ktorom býva Mate. Ive je najstarší, Mate najmladší z nich. A tak i domy. Ivanov dom je najstarší a najmenší, Franičov je novší, pod ním pivnica s velikánskymi sudmi, a Mateho je už celkom nový, s akýmsi nádychom luxusu, pravda ťažackého, sedliackeho.

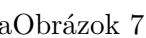
Franič dosť dávno vystavil svoj dom, ale ho nedohotovil. Vyzerá v ňom všetko akosi provizórne. Štyri steny zapáckané zhruba maltou, podlaha z dosiek, pod ktorou je spomenutá pivnica, ale povaly ešte...

²¹Validácii sa venujeme nižšie.

XML

Tento text, spolu s jeho metadátami, môžeme reprezentovať v XML nasledujúcim spôsobom:

```
<book>
  <title>Dom v stráni</title>
  <author>
    <name>Martin Kukučín</name>
    <dateOfBirth>1860</dateOfBirth>
    <dateOfDeath>1928</dateOfDeath>
  </author>
  <text>
    <paragraph>
      V stráni pod <place>Grabovnikom</place>, stáby prilepené...
    </paragraph>
    <paragraph>
      <character>Franič</character> dosť dávno vystavil svoj dom...
    </paragraph>
    <pb />
  </text>
</book>
```

Vizualizáciu štruktúry tohto dokumentu potom zachytáva 

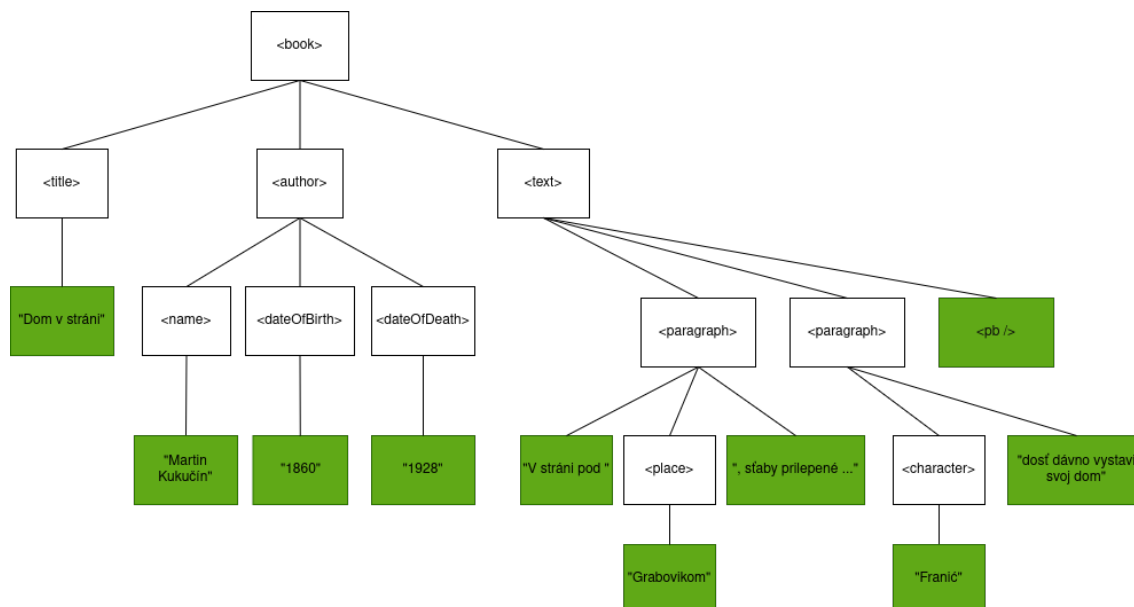
Komentáre

Sú akákoľvek časť dokumentu nachádzajúca sa medzi `<!--` a `-->`. Slúžia na dokumentáciu alebo vysvetlenie častí dokumentov. Parseery ich ignorujú a nemajú vplyv na štruktúru údajov:

```
<!-- Toto je komentár -->
```

Pokyny na spracovanie (Processing Instructions)

Informujú aplikácie, ako majú spracovať dokument alebo niektorú z jeho častí. Príkladom takýchto inštrukcií je tzv. deklarácia XML dokumentu:



Obrázok 7: Vizualizácia stromovej štruktúry XML dokumentu (prvky zvýraznené zelenou predstavujú listy stromu)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

Ak sa v dokumente nachádza²², tak musí byť umiestnená na jeho úplnom začiatku. Obsahuje informácie o kódovaní²³, verzii²⁴ a “standalone” stav dokumentu²⁵. Deklarácia slúži ako hlavička metadát, ktorá umožňuje parserom a procesorom správne interpretovať obsah dokumentu.

CDATA (*Character Data*)

predstavujú bloky textu, ktoré parsery neinterpretujú ako XML kód. Znaký ako ‘<’, ‘>’ sa v týchto sekciách teda považujú za bežné znaky, nie ako začiatok alebo koniec značiek. Užitočné je to v situáciách, keď by sa v analyzovanom texte nachádzali sekvencie, ktoré by parser štandardne indentifikoval ako indikácie XML prvkov, čomu chceme zabrániť. Zabezpečíme to tak, že ich ohraničíme na začiatku značkou “<![CDATA[” a na konci “]]>”.

²²Nie je to povinná súčasť XML dokumentov, ale obsahuje informácie, ktoré zjednodušujú ich spracovanie automatizačnými nástrojmi, takže je vhodné ju vždy uvádzať.

²³V tomto prípade ide o Unicode formu kódovania UTF-8, ktorej sme sa venovali v predchádzajúcej kapitole.

²⁴Verzia 1.0, definovaná v roku 1998, je najrozšírenejšou a odporúčanou verziou XML. Okrem nej existuje aj verzia 1.1, ktorá sa od predchádzajúcej verzie líši v niekoľkých ohľadoch. Tie tu však nebudeme uvádzať, keďže novšia verzia je málo rozšírená a jej špecifiká pre nás nie sú pre nás podstatné. Ďalšie verzie XML zatiaľ neexistujú.

²⁵Ide o informáciu, či je dokument závislý výhradne od informácií, ktoré sa v ňom nachádzajú (‘yes’) alebo nie (‘no’).

XML

Čokoľvek takto ohraňované sa považuje za “surový” text. Ak by sme teda chceli zakódovať úvodnú pasáž z predchádzajúcej sekcie (“Pokyny na spracovanie”) do XML, museli by sme to spraviť, napríklad, takto:

```
<section>
  <title>Pokyny na spracovanie (Processing Instructions)</title>
  <p>
    informujú aplikácie, ako majú spracovať dokument alebo niektorú z jeho častí.
    Príkladom takýchto inštrukcií je tzv. deklarácia XML dokumentu:
  </p>
  <![CDATA[<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>]]>
</section>
```

Menné priestory

v komplexných XML dokumnetoch, ktoré využívajú kombináciu viacerých slovníkov, môže dochádzať ku konfliktom názvov. Dve rôzne schémy môžu, napríklad, definovať element `<title>` s rôznym významom. Menné priestory poskytujú mechanizmus na predchádzanie takýmto nejednoznačnostiam prostredníctvom identifikácie pôvodu každého elementu alebo atribútu. V XML dokumente odlíšenie elementov s rovnakým názvom dosiahneme tak, že pred značku každého z nich pripojíme prefix zložený z názvu ich menného priestoru a dvojbodky. Teda, za predpokladu, že jeden variant elementu `<title>` pochádza z menného priestoru ‘a’ a druhý z priestoru ‘b’, môžeme ich v dokumente odlíšiť ako `<a:title>` a `<b:title>`.

Pre použitie takýchto prefixov však musíme v dokumente menné priestory zdefinovať prostredníctvom atribútu *xmlns* obsiahnutého buď v začiatočnej značke daného elementu alebo v značke niektorého z jeho rodičov²⁶ Deklarácia menného priestoru má syntax: `xmlns:prefix=“URI”`, pričom *URI* (Uniform Resource Identifier) nemusí nevyhnutne predstavovať existujúcu internetovú adresu.²⁷ V dokumente by použitie dvoch variant `<title>` mohlo vyzeráť nasledovne:²⁸

²⁶Typicky sa menné priestory definujú v koreňovom elemente dokumentu.

²⁷Často sa však ako *URI* používa adresa stránok, na ktorých sa nachádzajú informácie o danom mennom priestore, resp. o schéme, ktorá s ním je asociovaná.

²⁸`<a:title>` tu predstavuje “knižný titul” a `<b:title>` “nadpis kapitoly”.


```

<a:title xmlns:a="https://xml.namespaces.org/a"
  xmlns:b="https://xml.namespaces.org/b">
  <b:title>Malka</b>
  <text>
    <chapter>
      <b:title>Stretnutie</b:title>
      <p>Bol som vtedy v nočnej.</p>
      <p>Bol som v nočnej službe, aby ste ma lepšie rozumeli.</p>
      <p>...</p>
    </chapter>
  </text>
</a:title>

```

Štruktúra XML dokumentu

Po preskúmaní základných stavebných prvkov XML si predstavme pravidlá, ktoré určujú, aké ich kombinácie utvárajú dokument, ktorý zodpovedá XML štandardu.²⁹ Tieto pravidlá spadajú pod dve súvisiace, ale odlišné kategórie: *správna forma* a *validita*.

Obe kategórie zastrešujú požiadavky, ktorých splnenie je podmienkou toho, aby mohol byť dokument správne spracovaný softvérom, ale operujú na rôznych úrovniach. Mať správnu formu je minimálna požiadavka kladená na každý XML dokument; validita je striktnejšia obmedzenie, ktoré vyžaduje štrukturálnu konzistenciu dokumentu vzhľadom na určitý formálny model.

Správna utvorenosť (well-formedness)

Správne utvorený XML dokument je taký dokument, v ktorom sú všetky stavebné prvky použité v súlade so základnými syntaktickými pravidlami štandardu XML. Tie sa dajú zhrnúť do piatich bodov:³⁰

1. Dokument musí mať jeden a len jeden koreňový element, ktorý obsahuje všetky ostatné elementy v dokumente.

²⁹ *Extensible Markup Language (XML) 1.0 (Fifth Edition)*.

³⁰ Gulbransen et al., *Special Edition Using XML*, 2nd Edition.

XML

2. Každý element musí byť utvorený zo začiatkovej a koncovkej značky alebo musí mať podobu prázdneho elementu.
3. Všetky prvky musia byť správne vnorené
4. Všetky názvy elementov a atribútov musia dodržiavať XML konvencie pre pomenovania (t. j. nesmú sa začínať číslom, rozlišovanie veľkých a malých písmen, atď.)
5. Hodnoty atribútov sa dávajú do jednoduchých alebo dvojitéch úvodzoviek.

Príklad správne utvoreného dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <title>Dom v stráni</title>
  <author gender="M">Martin Kukučín</author>
  <year>1903</year>
</book>
```

Tento dokument je správne utvorený, pretože:

- má práve jeden koreňový element (<book>)
- všetky elementy sú správne vnorené a uzatvorené
- mená značiek a atribútov spĺňajú konvencie XML
- hodnoty atribútov sú v úvodzovkách

Príklad nesprávne utvoreného dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<author gender=M>Martin Kukučín
<book>
  <title>Dom v stráni<title>
  <year>1903</Year>
</book>
```

Tento dokument nie je správne utvorený, pretože:

- Nemá jeden a len jeden koreňový element
- Element <title> nie je správne uzavretý (je otvorený značkou <title>, ale namiesto </title> je nesprávne uzavretý opäť pomocou <title>)

- Element `<author>` nie je uzavretý.
- Hodnota atribútu `gender` nie je v údazovkách
- Koncová značka3 elementu3š3 `<year>` nekorešponduje so začiatočnou značkou (keďže v XML sa rozlišuje medzi veľkými a malými písmenami)

Validita

Validný XML dokument je taký dokument, ktorý je 1) správne utvorený a 2) zodpovedá formálnej gramatike alebo modelu definovanému prostredníctvom DTD (Document Type Definition), XML schéma (XSD) alebo RELAX NG.³¹ Tento model, bežne označovaný ako XML schéma, opisuje, aké elementy a atribúty sú povolené, v akom poradí a koľkokrát sa môžu vyskytovať, aké typy hodnôt môžu obsahovať, atď.

Validácia je v podstate záväzok medzi dokumentom a deklarovaným modelom. Zatiaľ čo požiadavka správnej utvorenosti dokumentu zabezpečuje, že je štruktúrovaný ako strom, jeho validita garantuje, že je tento strom správnym druhom stromu pre danú aplikáciu alebo oblasť formálne vymedzenú XML schémou. Všetky validné dokumenty teda musia byť dobre utvorené, ale nie všetky dobre utvorené dokumenty sú validné.

Pre ilustráciu validácie XML dokumentu si vezmeme nasledujúcu DTD schému:

```
<!DOCTYPE book [
  <!ELEMENT book (title, author)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
]>
```

Toto môžeme vyjadríť v bežnej reči v podobe nasledujúcich požiadaviek:

- element `<book>` musí obsahovať *najprv* element `<title>` nasledovaný elementom `<author>` a prostriedky elementy `<title>` a `<author>` musia obsahovať len text (PCDATA, resp. “parsed character data”)

Nasledujúci XML dokument je teda validný vzhľadom na uvedenú schéma schému:

³¹Pre bližšie oboznámenie sa s týmito spôsobmi, ako definovať XML model, pozri (DeRose, *The SGML FAQ Book*)

XML

```
<ixml version="1.0" encoding="UTF-8"?>
<book>
  <title>Dom v stráni</title>
  <author>Martin Kukučín</author>
</book>
```

Zatiaľ čo tento voči nej nie je validný (aj keď je dobre utvorený):

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  author>Martin Kukučín</author>
  <title>Dom v stránikonkrétneho</title>
</book>
```

Praktické výhody validácie XML sú obzvlášť významné pri kolaboratívnych a dlhodobých digitálnych projektoch. Vďaka tomu, že validácia zabezpečuje súlad dokumentov so spoločnou schémou, podporuje konzistentnosť medzi prispievateľmi a zabraňuje štrukturálnym chybám, ktoré by inak mohli zostať nepovšimnuté. Validácia ďalej umožňuje automatizovať pracovné postupy - napríklad transformáciu pomocou XSLT, publikovanie prostredníctvom platforiem ako TEI Publisher alebo integráciu do vyhľadávacích systémov - tým, že zaručuje predvídateľnú štruktúru a konzistentné používanie XML elementov. Validácia tiež uľahčuje interoperabilitu s inými nástrojmi a systémami, čím uľahčuje zdieľanie údajov

medzi inštitúciami alebo projektmi. Validácia v podstate funguje ako mechanizmus kontroly kvality, ktorý podporuje efektívnosť, spoľahlivosť a dlhodobú udržateľnosť postupov využívajúcich XML formát.

V tejto časťipoužitia sme predstavili základné črty XML formátu a validačných schém, ale nevenovali sme sa praktickým detailom toho, ako v skutočnosti prebieha proces validácie XML dokumentu. Pri tomto procese sa totiž musíme vysporiadať ešte napríklad aj s nasledujúcimi problémami: *The TEI Guidelines*³²

- ako procesor identifikuje validačnú schému (alebo schémy), ktorej alebo ktorým by mal určitý dokument zodpovedať?

32

- XML dokument môže byť uložený v množstve rôznych súborov operačného systému; ako by mal byť výsledný dokument z týchto častí vyskladaný?
- ako procesor interpretuje procedurálne inštrukcie, ktoré sa v dokumente nachádzajú?

Rôzne systémy na spracovanie XML môžu pristupovať veľmi odlišne k riešeniu týchto a ďalších praktických problémov, keďže tieto nie sú súčasťou samotnej XML špecifikácie.³³

Sémantická expresivita XML

Ako sme videli, fundamentálnou dátovou štruktúrou akéhokoľvek XML dokumentu je strom, hierarchická štruktúra, v ktorej každý element (alebo „uzol“) môže obsahovať podriadené prvky, ktoré tvoria vnorené vrstvy. To nám umožňuje reprezentovať rôzne konceptuálne vzťahy odrážajúce logickú, textovú a sémantickú organizáciu analyzovaného obsahu.

Vzťah časti a celku

Známy aj ako *meronymia*, je jedným z najprirodzenejších spôsobov použitia vnorenia XML elementov, pričom vyjadruje to, ako menšie jednotky spolu utvárajú väčší celok. Pre ilustráciu si vezmeme knihu rozdelenú na kapitoly a odseky:

```
<book>
  <chapter>
    <title>Zakladajú spolok Rovnosť</title>
    <paragraph>
      Na veľkých visacích hodinách v jedálni U barana
      odbila jedna po polnoci...
    </paragraph>
  </chapter>
  <chapter>
    <title>Treba ukázať príklad</title>
    <paragraph>
      Len čo prišiel Landík domov a zapálil lampu,
```

³³Pozri *The TEI Guidelines* pre obecnější náčrt riešení týchto problémov. Pre detailnejší a technickejší pohľad pozri *RELAX NG Home Page*

XML

```
        už mu niekto zaklopal...  
    </paragraph>  
  </chapter>  
</book>
```

Vzťahy medzi elementami <book>, <chapter>, <title> a <paragraph> tu odrážajú fyzickú štruktúru knihy, t.j., že každá kapitola je súčasťou knihy a každý odsek je súčasťou príslušnej kapitoly.

Typologické vzťahy

hierarchické vzťahy stelesnené v XML strome môžu ďalej reprezentovať to, že určité entity sú inštanciami nejakej kategórie. Vezmime si napríklad nasledujúci zoznam postáv nejakého románu:

```
<characters>  
  <protagonist>Anna</protagonist>  
  <antagonist>Juraj</antagonist>  
  <support>Mária</support>  
</characters>
```

Každá postava (reprezentovaná značkami <protagonist>, <antagonist> alebo <support>) je typ osoby vystupujúcej v príbehu. Nadradený prvok <characters> definuje kategóriu, ktorej sú vnorené elementy inštanciami.

Časové a sekvenčné vzťahy

Hoci sú stromy XML prirodzene hierarchické, poradie súrodeneckých prvkov môže predstavovať aj časovú alebo logickú postupnosť. Pre ilustráciu si vezmime p nasledujúcu posutpnosť záznamov v denníku:

```
<diary>  
  
  <entry date="1939-09-01">Začala vojna.</entry>
```

```
<entry date="1939-09-20">Dnes som dostal povolávací rozkaz...</entry>
</diary>
```

Prvky `<entry>` tu nie sú len časťami `<diary>`; ich poradie môže odrážať jednak poradie denníkových záznamov, ako aj plynutie času a vývoj udalostí.

Deskriptívne vzťahy (atribúcia a anotácia)

XML umožňuje prvkom niesť popisy alebo atribúty iných prvkov, ako napríklad v nasledujúcej štruktúre, ktorá kóduje vlastnosti nejakej osoby:

```
<person>
  <name>Terézia Vansová</name>

  <birthDate>1857-4-18</birthDate>
  <occupation>Spisovateľka</occupation>
</person>
```

kde každý podradený prvok opisuje iný aspekt osoby identifikovanej koreňovým prvkom.

Referenčné vzťahy

Pri kódovaní je niekedy potrebné zaznamenať prepojenie prvkov, ktoré spolu logicky súvisia, ale nie sú štrukturálne vnorené, ako napríklad prepojenie poznámky pod čiarou s úryvkom, ktorého sa týka. Štruktúra XML dokumentov je síce vo svojej podstate hierarchická, ale toto obmedzenie je možné obísť pomocou odkazov implementovaných prostredníctvom XML atribútov.

V nasledujúcom príklade vidíme, že `<note>` síce nie je potomkom `<paragraph>`, ale odkazuje naň pomocou atribútu `“target”`, ktorého hodnota obsahuje identifikátor relevantného paragrafu, pričom ten je mu pridelený prostredníctvom atribútu `“id”`.

```
<paragraph id="p1">
  My ale, ktorí sa tej katastrofy nedočkáme, obráťme našu
  pozornosť k národnostnej otázke a k makovým opekancom.
</paragraph>
```

```
<note target="#p1">
```

Trochu sa síce opozdila táto besednica, ale opekance s makom by vari ešte i teraz nikto neohrdil.

```
</note>
```

Kontextové a diskurzívne vzťahy

V literárnych a historických materiáloch je často dôležité ukázať, ako je nejaký výrok prezentovaný v diskurzívnom kontexte - napríklad uviesť hovorcu repliky v dialógu alebo to, že daný riadok predstavuje verš básne.

```
<div>
```

```
  <head>Prvý obraz</head>
```

```
  <utterance speaker="Bernardo">
```

Kto tam?

```
</utterance>
```

```
  <utterance speaker="Francisco">
```

A ty si kto? Povedz heslo! Stoj!

```
  <utterance speaker="Bernardo">
```

Nech žije kráľ!

```
</utterance>
```

```
  <utterance speaker="Francisco">
```

Bernardo?

```
</utterance>
```

```
</div>
```

V predchádzajúcom príklade má každá z replík - reprezentovaných značkou `<utterance>` - nastavený atribút "speaker", ktorého hodnota označuje jej hovorcu.

Redakčné alebo interpretačné vrstvenie

Digitálne edície literárnych diel, si často vyžadujú interpretačných vzťahov - napríklad označenie redakčných korekcií, neistého čítania alebo viacerých verzií úryvku. XML umožňuje modelovať ich prostredníctvom vnorených prvkov alebo atribútov. Pre príklad

si vezmiem situáciu, keď chce editor zaznamenať opravu chyby sa nachádza v originálnom texte:

<p>

Nuž, bračekovci, verte alebo nie, ale ja som sa vtedy cítil ako
 <choice><orig>mys</orig><corr>myš</corr></choice> v kyslom
 mlieku.

</p>

Pôvodná a opravená verzia tu koexistujú v jednej redakčnej štruktúre.

Táto séria príkladov ukazuje, že prostredníctvom formátu XML sme schopní vyjadriť oveľa viac typov vzťahov než len vzťah medzi celkom a jeho časťami. Umožňuje nám ukázať, ako veci súvisia - štrukturálne aj koncepčne. Vďaka tomu je tento formát obzvlášť vhodný pre disciplíny, ako je literatúra, história a lingvistika, kde význam často závisí od vzťahov medzi ľuďmi, textami, udalosťami a interpretáciami.

TEI (**Text Encoding Initiative**)

Iniciatíva pre kódovanie textu (Text Encoding Initiative, TEI) je v digitálnych humanitných vedách široko prijatý štandard na reprezentáciu textov v digitálnej forme pomocou XML. Základom tohto štandardu je schéma TEI XML, komplexný a prispôsobiteľný model sémantického kódovania textových javov - od bibliografických metadát, cez rôzne štrukturálne aspekty literárnej produkcie, až po komplexné varianty rukopisov.

Formálne je schéma TEI definovaná prostredníctvom už vyššie spomínaných jazykov RELAX NG, DTD alebo W3C XML Schema, čo umožňuje validáciu dokumentov voči modelom kompatibilných s TEI. Pravidlá tejto schémy opisujú nielen to, ktoré XML

Elementy sa môžu v dokumentoch používať (napr. <div>, <p>, <persName>, <date> atď.), ale aj to, ako môžu byť vnorené, aké atribúty môžu obsahovať a v akom poradí sa môžu vyskytovať.

TEI sa vyznačuje schopnosťou reprezentovať komplexné redakčné a interpretačné informácie, ako sú textové varianty, anotácie, štrukturálne hierarchie a sémantické prvky.

Napríklad vydanie slovenského románu zakódované v TEI môže obsahovať nielen štruktúru kapitol a odsekov, ale aj označenie mien historických postáv (<persName>), miest (<placeName>), dátumov (<date>) a edičných poznámok (<note>).

TEI však nie je len súborom XML elementov, ale predstavuje flexibilný rámec prispôsobiteľný špecifickým potrebám konkrétneho výskumného projektu. To sa dosahuje predovšetkým prostredníctvom mechanizmu, ktorý využíva špecifikáciu ODD (One Document Does it all) na definovanie toho, ktoré prvky, atribúty a moduly sa do upravenej schémy zahrnú, vylúčia alebo zmenia. Používatelia môžu pridávať nové elementy, meniť modely obsahu alebo obmedzovať používanie určitých značiek, a to všetko pri zachovaní kompatibility s validačnými nástrojmi a dokumentačnými systémami.³⁴

TEI Moduly: Funkčné stavebné prvky schémy

Špecifikácia TEI, nachádzajúca sa v *The TEI Guidelines*³⁵ obsahuje definície niekoľkých stoviek elementov a atribútov. Každá definícia pritom pozostáva z v bežnom jazyku vyjadreného opisu definovanej entity, jej formálnej deklarácie vyjadrenej prostredníctvom kombinácie špecializovaného XML slovníka a prvkov pochádzajúcich z jazyka schémy RELAX NG a praktických príkladov použitia daného elementu alebo atribútu.

poskytuje štrukturálny rámec najvyššej úrovne pre každý dokument kódovaný v TEI. Definuje základné prvky, ktoré sú potrebné na vytvorenie súboru v súlade so schémou, a funguje ako obal pre metadáta (hlavička TEI) aj textový obsah. Tento modul v podstate ukotvuje dve hlavné zložky dokumentu: opisné metadáta a kódovaný text.

Table 1: Kľúčové elementy modulu `tei`

Element	Popis
<TEI>	koreňový element každého TEI dokumentu
<teiHeader>	obsahuje metadáta dokumentu, ako napríklad bibliografické údaje alebo záznam revízií
<text>	Obsahuje štruktúrovaný textový obsah (telo, predná strana, zadná strana atď.).

³⁴Pre bližšie oboznámenie sa s tým, ako funguje systém modifikácie TEI schémy, pozri: *Getting Started with P5 ODDs*.

³⁵

Minimálny dokument využívajúci prvky z tohto modulu vyzerá takto:

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
    <!-- metadáta o texte -->
  </teiHeader>
  <text>
    <!-- obsah reprezentovaného textu -->
  </text>
</TEI>
```

Modul header poskytuje prostriedky pre zaznamenávanie štruktúrovaných metadát dokumentov. Všetky tieto informácie sa uvádzajú pod elementom `<teiHeader>`, ktorý musí byť prítomný v každom validnom TEI dokumente.

Element	Popis
<code><fileDesc></code>	reprezentuje bibliografické a publikačné informácie
<code><encodingDesc></code>	Podrobnosti o spôsobe kódovania textu vrátane redakčných a kódovacích zásad
<code><profileDesc></code>	Poskytuje kontextuálne a deskriptívne informácie o obsahu.

uvádzame v.tbl. 5³⁶ | `<revisionDesc>` | Záznam zmien vykonaných v súboroch v priebehu času |

: Kľúčové elementy modulu **header** {#tbl:headermod-elements__tbl}

Ilustráciu použitia elementov z tohto modulu ilustruje tento fragment XML dokumentu:

```
<tei:teiHeader>
  <tei:fileDesc>
    <tei:titleStmt>
      <tei:title>
        Adam Šangalav
```

³⁶Pre detailné oboznámenie sa s obsahom jednotlivých modulov pozri *The TEI Guidelines*

```

</tei:title>
<tei:author ref="viaf:45478401">
  Nádaši-Jégé, Ladislav (1866 - 1940)
</tei:author>
</tei:titleStmt>
<tei:publicationStmt>
  <tei:p/>
</tei:publicationStmt>
<tei:sourceDesc>
  <tei:bibl type="printSource">
    <tei:author>
      Nádaši-Jégé, Ladislav
    </tei:author>
    <tei:title>
      Adam Šangala
    </tei:title>
    <tei:pubPlace>
      Bratislava
    </tei:pubPlace>
    <tei:publisher>
      Tatran
    </tei:publisher>
    <tei:date>
      1970
    </tei:date>
  </tei:bibl>
</tei:fileDesc>
</tei:teiHeader>

```

Modul core obsahuje súbor elementov, ktoré sa používajú v takmer všetkých TEI dokumentoch. Ide o všeobecné, opakovane použiteľné a sémanticky neutrálne elementy, takže sú vhodné na kódovanie širokej škály textových javov. Vďaka ich flexibilita a všadeprítomnosti je **core** modul základom mnohých ďalších špecializovaných modulov.

Tento modul je automaticky zahrnutý do každého prispôsobenia TEI, pokiaľ nie je explicitne vylúčený, a jeho prvky sú k dispozícii v metadátoch (v `<teiHeader>`) aj v textovom obsahu (v `<text>`).

Table 3: Kľúčové elementy modulu **core**

Element	Popis
<code><p></code>	Odsek - používa sa na označenie blokov prózy
<code><ab></code>	Anonymný blok - generický kontajner bez zreteľnej štruktúry
<code><head></code>	Nadpis alebo názov časti alebo bloku
<code><hi></code>	Zvýraznený text
<code><note></code>	Poznámky pod čiarou, poznámky na konci knihy, marginálie alebo edičné poznámky.
<code><ref></code>	Generický odkaz (môže byť interný alebo externý)
<code><list></code> / <code><item></code>	Zoznam / položka zoznamu
<code><quote></code>	Citovaný materiál - priama alebo nepriama citácia.
<code><seg></code>	Generický segment pre anotácie v riadku.
<code><lb></code> / <code><pb></code>	Zalomenie riadku / zalomenie strany - na znázornenie rozloženia textu alebo pôvodnej štruktúry.

Príklad použitia niektorých elementov z modulu **core**:

```
<div type="chapter" xml:id="ch1">
  <head>Začiatok cesty</head>
  <p>
    <hi rend="italic">Bol raz jeden muž</hi>, ktorý sa rozhodol opustiť svoj domov.
    <note place="margin">Typický začiatok rozprávky.</note>
  </p>
</div>
```

Modul textstructure poskytuje základný rámec pre organizáciu tela textu v dokumente prostredníctvom jeho členenia na kapitoly, sekcie, úvody a ďalšie štrukturálne jednotky. Elementy modulu umožňujú neobmedzené hierarchické vnáranie prvkov, vďaka čomu možno reprezentovať architektúru arbitrárne komplexných diel.

Table 4: Kľúčové elementy modulu `textstructure`

Element	Popis
<code><text></code>	kontajner pre textový obsah dokumentu
<code><body></code>	hlavný obsah diela
<code><div></code>	Generický kontajner pre logické členenie, ako sú kapitoly alebo oddiely.
<code><front></code>	Obsahuje predsádky, predhovory, venovania, titulné strany atď.
<code><back></code>	Koncový materiál, ako sú prílohy alebo edičné poznámky.

Príklad použitia elementov z modulu `textstructure` na reprezentáciu titulnej strany diela a členenia na kapitoly:

```

<tei:text>
  <tei:front>
    <tei:div type="titlepage">
      <tei:head>
        Adam Šangala
      </tei:head>
    </tei:div>
  </tei:front>
  <tei:body>
    <tei:div type="chapter">
      <tei:head>
        1
      </tei:head>
      <tei:p>
        Začiatkom XVII. storočia bývalo obyvateľstvo úzkych dolín oravskej Magury ...
      </tei:p>
      ...
    <tei:div type="chapter">
      <tei:head>
        1

```

```

</tei:head>
<tei:p>
    Druhé ráno sa úzkou dolinou, vedúcou medzi vysokými ...
</tei:p>
...
</tei:div>
</tei:body>
</tei:text>

```

Špecifikácia TEI obsahuje ďalších osemnásť modulov, ktorými sa tu nebudeme podrobnejšie zaoberať. Celkový zoznam modulov, s ich stručnou charakterizáciou,

Table 5: Zoznam všetkých TEI modulov

Meno modulu	Obsah modulu
tei	celková štruktúra dokumentu.
header	metadáta popisujúce zdroj, vznik a redakčnú históriu dokumentu
core	univerzálne prvky na označovanie základných textových štruktúr
textstructure	hierarchické usporiadanie obsahu dokumentu
gaiji	reprezentácia neštandardných znakov alebo glyfov
verse	básnických textov
drama	dramatické texty
spoken	prejavy hovoreného jazyka
cmc	počítačom sprostredkovaná komunikácia
dictionaries	lexikálne zdroje, ako sú slovníky, glosáre a slovné zoznamy
msdescription	fyzické a kodikologické vlastnosti rukopisov
transcr	transkripčia hovoreného, posunkového alebo iného neštandardného jazyka
textsrit	kritika textov vrátane variantov, aparátu a redakčných zásahov
namesdates	mená osôb, organizačné jednotky, názvy miest a dátumy
figures	vkladanie a opisovanie grafických materiálov
corpus	zoskupenie viacerých dokumentov do štruktúrovanej zbierky alebo korpusu

Meno modulu	Obsah modulu
linking	vzťahy a prepojenia medzi časťami dokumentu alebo medzi viacerými dokumentmi
iso-fs	lingvistickú anotáciu a analýzu
nets	kódovanie sietí vzťahov, asociácií alebo závislostí v dokumentoch
certainty	vyjadrenie stupňa istoty, presnosti a zodpovednosti
tagocs	dokumentácia a popis značkovacích postupov, schém a používania elementov

ELTeC úprava TEI

ELTeC (European Literary Text Collection) je iniciatíva vyvinutá v rámci aktivity COST „Distant Reading for European Literary History“ (CA16204).³⁷ Jej hlavným cieľom je vytvoriť viacjazyčnú, žánrovo špecifickú zbierku európskych literárnych textov - konkrétne románov - v rôznych jazykoch a obdobiach, s jednotným kódovaním a metadátovými štandardmi. Technicky ide o prispôsobenie špecifikácie TEI za účelom interoperability, porovnateľnosti a strojovej čitateľnosti naprieč národnými literárnymi tradíciami.

ELTeC predstavuje zjednodušenú a obmedzenú podmnožinu úplnej schémy TEI, obohatenú o súbor spoločných konvencií a postupov. Hoci zachováva kompatibilitu s originálnou schémou, zámerne obmedzuje a štandardizuje niektoré z jej aspektov. Jedným z charakteristických znakov je dôraz na odľahčené kódovanie. Ide tu skôr o zachytenie základných štruktúrnych a bibliografických informácií a nie o podrobnú jazykovú alebo interpretačnú anotáciu. Tento prístup vyvažuje výpočtovú jednoduchosť s redakčnou transparentnosťou, vďaka čomu je výsledná zbierka veľmi dobre použiteľná na dištančné čítanie, štylistickú analýzu a medzijazykové porovnávanie.

ELTeC úprava TEI je implementovaná ako ODD špecifikácia, ktorá definuje povolené TEI elementy, atribúty a obmedzenia ich použitia. Vo všeobecnosti možno zhrnúť princípy týchto modifikácií nasledovne:

1. **Uniformita:** presadzovanie jednotného kódovania v korpusoch z rôznych jazykov a národných tradícií.

³⁷„Action CA16204“.

2. **Jednoduchosť:** obmedzenie používania zložitých alebo voliteľných konštrukcií TEI v prospech malého, jasne zdokumentovaného súboru.
3. **Strojová čitateľnosť:** umožňuje automatizované spracovanie pre dištančné čítanie, štýlometriu a ďalšie digitálne metódy.
4. **Minimalizmus:** dôraz na základné metadáta a prvky textu a obmedzenie nadmerného značkovania.

Kľúčové zložky ELTeC ODD

Celková štruktúra dokumentu Rámec kódovania ELTeC zavádza jednotnú štruktúru dokumentov vo všetkých textoch v korpuse. Model sa riadi obmedzenou, ale expresívnou podmnožinou špecifikácie TEI a je štruktúrovaný okolo niekoľkých základných hierarchických prvkov:

- Každý dokument musí obsahovať práve jeden element `<text>`.
- Telo textu je členené do kapitol a podkapitol prostredníctvom elementu `<div type="chapter">`.
- Kapitoly sú rozdelené do ďalej neštruktúrovaných paragrafov reprezentovaných elementom `<p>` (nevyžadujú sa vnorené vety ani členenie na riadky).

Štandardizácia TEI hlavičky `<teiHeader>` je povinná zložka každého ELTeC dokumentu a obsahuje elementy v nasledujúcej štruktúre:

- `<fileDesc>`
 - `<titleStmt>`: obsahuje názov a jeden alebo viacero elementov `<author>`.
 - `<publicationStmt>`: štandardizované informácie o licenciách a vydavateľoch (typicky Creative Commons).
 - `<sourceDesc>`: bibliografické informácie o zdrojovom vydaní použitom na kódovanie.
- `<profileDesc>`
 - `<langUsage>`: zoznam ISO kódov jazykov použitých v texte.
 - `<textDesc>`: uvádza pohlavie autorstva, časové obdobie vzniku, kánonicitu a veľkosť diela

- `<revisionDesc>`: sledovanie histórie zmien dokumentu

Značkovanie obsahu Ako sme už uviedli, značkovanie obsahu je zámerne odľahčené pre účely čitateľnosti a počítačového spracovania vo viacerých jazykoch a literárnych tradíciách. Dôraz sa kladie na štruktúralnu prehľadnosť, základnú sémantickú anotáciu a minimalizáciu interpretačných zásahov. Nižšie uvádzame najčastejšie používané elementy.

`<p>` - *Paragraf*

- Základná jednotka prózy.
- Vyžaduje sa vo vnútri každého `<div type="chapter">`.
- Obsahuje priebežný text kódovaného románu.
- Môže obsahovať inline prvky, ako sú `<hi>`, `<name>`, `<note>` a `<foreign>`.

`<div type="chapter">` — *Štruktúralne členenie*

- Používa sa na rozdelenie románu na kapitoly alebo iné štruktúralne jednotky.
- Musí byť vnorený do `<body>` elementu.
- Obsahuje sekvenciu paragrafov reprezentovaných elementami `<p>`.

`<head>` — *Nadpis*

- Nepovinný, ale odporúčaný element uvádzaný na začiatku každého členenia prostredníctvom `<div>`.
- Predstavuje názov alebo číslo kapitoly alebo oddielu.

`<hi>` - *Zvýraznený text*

- Používa sa na text, ktorý sa zobrazuje špeciálnym spôsobom .
- Typ zvýraznenia (t.j., kurzíva, tučné písmo atď.) sa ovláda pomocou atribútu `@rend`.

`<note>` - *Redakčná alebo vysvetľujúca poznámka*

- Používa sa zriedkavo na poznámky pod čiarou, marginálie alebo redakčné spresnenia.
- Môže obsahovať atribúty ako `@place` (napr. „margin“, „foot“) alebo `@type`.

`<name>` — *Proper Names*

- Označuje pomenované entity, ako sú osoby, organizácie alebo miesta.

- Pomáha pri rozpoznávaní a prepájaní pomenovaných entít.

<foreign> - *Text v cudzom jazyku*

- Identifikuje úseky textu v inom jazyku, ako je hlavný jazyk dokumentu.
- Často obsahuje atribút **@xml:lang** na určenie jazyka daného úseku.

<pb> - *Zlomky strán* (voliteľné)

- Používa sa, keď je dôležité zachovať stránkovanie zdroja.
- Pomocou atribútu **@n** sa identifikuje číslo strán.

<gap> - *Vynechaný alebo nečitateľný text* (voliteľné)

- Označuje chýbajúcu časť textu, napr. v dôsledku poškodenia zdroja alebo redakčného vynechania.

GIT - distribuovaný systém na kontrolu verzií

S rastúcou zložitou a spoluprácou na projektoch digitálnych humanitných vied je čoraz dôležitejšie sledovať zmeny, spravovať viaceré príspevky a udržiavať prehľadnú históriu vyvíjajúcich sa dokumentov. Systémy na tvorbu verzií - nástroje určené na zaznamenávanie, porovnávanie a správu revízií v priebehu času - ponúkajú spoľahlivé riešenie týchto problémov. Systémy ako Git, pôvodne vyvinuté pre vývoj softvéru, našli prirodzené miesto v pracovných postupoch v humanitných vedách a podporujú spoločné úpravy, prehľadnú históriu revízií a reprodukovateľný výskum. Táto kapitola predstavuje princípy a postupy riadenia verzií so zameraním na to, ako ich možno efektívne uplatniť v textovej vede, digitálnych edíciách a tímovom humanitnom výskume.

Webové technológie

World Wide Web, bežne označovaný ako “web”, je rozsiahly a vzájomne prepojený systém digitálnych dokumentov a zdrojov prístupných prostredníctvom internetu. Umožňuje používateľom prechádzať informácie pomocou hypertextových odkazov - klikateľných spojení medzi jednotlivými časťami obsahu - a vytvára tak dynamické a nelineárne prostredie, ktoré sa líši od tradičných médií. Technológiu, ktorú pôvodne navrhol Tim Berners-Lee v roku 1989 ako spôsob zdieľania výskumných dokumentov medzi inštitúciami, sa odvtedy vyvinul do všadeprítomnej platformy na komunikáciu, zdieľanie vedomostí, obchodovanie a kultúrnu produkciu.

Web vo svojej podstate funguje prostredníctvom kombinácie štandardizovaných technológií. Webové prehliadače interpretujú jazyk HTML (HyperText Markup Language) na zobrazenie textového a multimediálneho obsahu, CSS (Cascading Style Sheets) na vizuálnu prezentáciu tohto obsahu a JavaScript na umožnenie interaktivity. Súbory obsahujúce zdrojové kódy v týchto jazykoch sú umiestnené na webových serveroch a prístupuje sa k nim pomocou protokolov HTTP alebo HTTPS, ktoré sprostredkujú výmenu údajov medzi klientmi a servermi.

Dištinktívnym znakom webových technológií je ich otvorenosť a decentralizácia. Ktokoľvek, kto má základné technické znalosti, môže publikovať obsah, ktorý je okamžite prístupný celosvetovému publiku. Pre vedcov, tvorcov a používateľov predstavuje web nielen médium na konzumáciu, ale aj nástroj na publikovanie, spoluprácu a experimentovanie. V kontexte digitálnych humanitných sa web používa nielen ako prostriedok na publikovanie statického obsahu, ale ako dynamická platforma na vytváranie interaktívnych vedeckých prostredí založených na dátach. Od digitálnych edícií historických textov až po interaktívne vizualizácie literárnych sietí web umožňuje projektom osloviť širšie publikum a zapojiť používateľov spôsobom, akým to tradičné médiá nedokážu.

Podstatou DH je prepojenie počítačových metód a humanitného výskumu. Webové technológie slúžia ako rozhranie a infraštruktúra tohto priesečníka. Prostredníctvom nich možno vytvárať flexibilné a prístupné platformy, ktoré prezentujú texty, metadáta a analytické nástroje integrovaným a užívateľsky prívetivým spôsobom. Či už ide o poskytovanie online korpusu, kritického aparátu alebo vizualizácie geopriestorových údajov, vývoj webových stránok umožňuje spojenie interpretačnej vedy s dizajnom a interaktivitou.

Okrem toho otvorený a kolaboratívny étos webu odráža princípy mnohých DH projektov. Úložiská s verziovacími systémami, modulárny kód, otvorené formáty údajov a opakovane použiteľné komponenty podporujú transparentnosť a opakované použitie - podporujú vedecké ekosystémy, ktoré sú udržateľné a rozšíriteľné. Schopnosť prepojiť zdroje, vkladať sémantické údaje a prispôbovať rozhrania rôznym potrebám používateľov posilňuje vedeckú hodnotu a verejný vplyv digitálnej práce.

Stručne povedané, webové technológie nie sú v digitálnych humanitných vedách okrajovou záležitosťou, ale v ich centre. Keďže sa očakáva, že čoraz viac výstupov výskumu bude prístupných, navigovateľných a otvorených, digitálni humanisti čoraz viac potrebujú nielen tradičné kompetencie, ale aj základné praktické vedomosti o tom, ako web funguje.

Na vytvorenie účinných a užívateľsky prívetivých rozhraní pre humanitný obsah je nevyhnutné pochopiť fundamentálne webové technológie a ich vzájomné prepojenie. Hoci môžu byť údaje v mnohých projektoch zakódované v XML formáte, konečná prezentačná vrstva, teda to, ako sú dáta prezentované užívateľom, sa zvyčajne spolieha na tri technológie: HTML, CSS a JavaScript.

HTML (HyperText Markup Language) je jazyk používaný na štruktúrovanie obsahu na webe. Definuje elementy, ktoré tvoria webovú stránku - nadpisy, odseky, zoznamy, odkazy, obrázky tabuľky atď. - pomocou systému značiek, ktoré opisujú sémantickú úlohu každej časti obsahu. Pre digitálnych humanistov je oboznámenie sa s HTML nevyhnutné nielen na vytváranie alebo prispôbovanie webových rozhraní, ale aj na porozumenie tomu, ako sa bohato kódované dokumenty (ako napríklad TEI-XML) nakoniec transformujú na čitateľné a navigovateľné digitálne texty. Hoci HTML nevyjadruje komplexné vedecké metadáta alebo textové nuansy tak efektívne ako TEI, vyniká pri vykresľovaní štruktúrovaných informácií na verejnú prezentáciu, čo z neho robí dôležité premostenie medzi surovými dátami a koncovým užívateľom.

CSS (Cascading Style Sheets) je jazyk používaný na ovládanie vzhľadu a rozloženia obsahu HTML dokumentov. Zatiaľ čo HTML poskytuje štruktúru webovej stránky, CSS určuje, ako táto štruktúra vyzerá - stanovuje pravidlá pre písma, farby, medzery, zarovnanie a responzivitu užívateľských rozhraní voči displejom rôznych veľkostí. V projektoch digitálnych humanitných vied zohráva CSS dôležitú úlohu pri zlepšovaní čitateľnosti, zdôrazňovaní textových prvkov a udržiavaní koherentného vizuálneho dizajnu, ktorý odráža vedecký zámer edície alebo korpusu. Či už ide o štylizáciu diplomatického prepisu alebo rozlišovanie vrstiev anotácií, CSS umožňuje humanistom formovať vizuálnu rétoriku ich digitálnej práce bez toho, aby zasahovali do pôvodných dát.

JavaScript je výkonný skriptovací jazyk, ktorý do webových stránok prináša interaktivitu a dynamické správanie. Na rozdiel od HTML a CSS, ktoré definujú štruktúru a vzhľad dokumentov, vďaka JavaScriptu dokáže programátor reagovať na akcie používateľa, manipulovať s obsahom v reálnom čase a integrovať funkcie založené na údajoch. V digitálnych humanitných vedách sa JavaScript často používa na implementáciu funkcií, ako sú interaktívne mapy, rozvinuteľné poznámky pod čiarou, fazetové vyhľadávanie a dynamické vizualizácie. Vďaka svojej flexibilitě je nepostrádateľný pri vytváraní prieskumných rozhraní, ktoré umožňujú používateľom aktívne pracovať s digitálnymi textami, anotáciami alebo súbormi údajov. JavaScript síce zvyšuje zložitosť, ale umožňuje takú úroveň interakcie používateľov, akú statické stránky nedokážu dosiahnuť.

HTML

Predtým, než sa začneme zaoberať technickými aspektmi HTML, by sme sa mali oboznámiť s ideou *hypertextu*, ktorá je jedným z fundamentálnych princípov digitálnych médií a transformujúcim rámcom pre to, ako chápeme, štruktúrujeme a pracujeme s informáciami. V jadre hypertext predstavuje systém textových prepojení realizovaných prostredníctvom odkazov, ktoré umožňujú čitateľom nelineárne prechádzať medzi dokumentmi, sekciami alebo súvisiacim obsahom. Na rozdiel od tradičných tlačенých textov, ktoré vedú čitateľov prostredníctvom pevného, sekvenčného rozprávania, hypertext otvára mnohosmerovú oblasť významov. To odráža aj spôsob, akým ľudia často myslia a učia sa, vytváraním spojení medzi pojmami a sledovaním významových nitiek. Hypertext mení čítanie na prieskumný proces, kde každý odkaz otvára potenciálnu novú cestu skrz

informačný priestor. Podporuje viacúrovňové porozumenie, vzájomné odkazovanie a uľahčuje integráciu rôznych materiálov do ucelenej interaktívnej skúsenosti. Hypertext zásadne zmenil spôsob štruktúrovania, prístupu a interpretácie informácií v digitálnom prostredí, či už sa používa vo vzdelávacích platformách, technickej dokumentácii alebo online encyklopédiách.

Myšlienka hypertextu vznikla ešte pred moderným internetom a má korene vo víziách organizácie informácií z polovice 20. storočia. Jeden z prvých a najvplyvnejších konceptov pochádza od Vannevara Busha, ktorý vo svojej eseji „Ako môžeme myslieť“¹ z roku 1945 predstavil zariadenie s názvom *Memex* - teoretický stroj, ktorý by používateľom umožnil vytvárať a sledovať asociatívne trasy prepojených dokumentov. Táto vízia inšpirovala neskorší vývoj v oblasti výpočtovej techniky, najmä Teda Nelsona, ktorý v 60. rokoch 20. storočia vymyslel pojem „hypertext“ a pracoval na prvých systémoch, ako bol projekt Xanadu, ktorý sa snažil zaviesť nelineárne, vzájomne prepojené systémy písania.² Praktická realizácia hypertextu prišla s nástupom osobných počítačov a grafických používateľských rozhraní v 80. rokoch 20. storočia a vyvrcholila vytvorením World Wide Webu Timom Bernersom-Lee v roku 1989. Jeho vynález webového prehliadača, adres URL a HTML priniesol hypertext do každodenného používania. Odvtedy sa hypertext stal základným pojmom digitálnej komunikácie, ktorý formuje spôsob, akým píšeme, čítame a organizujeme vedomosti prakticky vo všetkých oblastiach.

Jednou z charakteristických črt humanitných zdrojov je ich intertextualita, teda spôsob, akým texty odkazujú, odrážajú, reagujú alebo nadväzujú na iné texty v čase a kontexte. Či už ide o literárny román čerpajúci z klasickej mytológie, historický dokument citujúci právne precedensy alebo filozofické pojednanie, ktoré nadväzuje na predchádzajúcich mysliteľov, materiály z humanitných vied sú zriedkavo v sebe uzavreté nezávislé monády. Význam takýchto diel často vyplýva nielen z ich vnútornej štruktúry, ale aj zo vzťahov s inými textami, tradíciami a kultúrnymi artefaktmi. Rozpoznávanie a reprezentovanie týchto komplexných sietí odkazov je jednou z ústredných úloh humanitných vied.

Hypertext tak ponúka vhodný rámec na reprezentáciu intertextuality, ktorá je taká dôležitá pre humanitný výskum. Tam, kde sa tradičné tlačené formáty obmedzujú na poznámky pod čiarou, bibliografie a nepriame odkazy, hypertext umožňuje priame, dynamické prepojenia medzi textami - umožňuje používateľom plynulý prechod z jedného

¹Vannevar Bush, *As We May Think*.

²Nielsen, *Multimedia and Hypertext*.

dokumentu do druhého alebo z úryvku do jeho komentára, zdroja alebo variantu. V projektoch digitálnych humanitných vied nie je hypertext len navigačným uľahčením, ale stáva sa metodologickým nástrojom na mapovanie dialogickej povahy textov, zviditeľňuje vrstvy citácií, adaptácií a alúzií, ktoré sú základom ich tvorby a interpretácie. Tým, že hypertext vkladá intertextuálne vzťahy priamo do štruktúry digitálnych edícií, archívov alebo korpusov, umožňuje čitateľom zaoberať sa materiálmi z oblasti humanitných vied spôsobom, ktorý bezprostredne odráža bohatu prepletenú povahu samotných zdrojov.

Ako sme už uviedli v krátkej historickej exkurzii vyššie, model hypertextu našiel svoju konkrétnu technickú realizáciu v jazyku HTML (HyperText Markup Language) vytvorenom Johnom Berners-Leeom. HTML je základným jazykom webu, ktorý je určený na štruktúrovanie a prepojenie dokumentov prostredníctvom súboru značiek. Z technického hľadiska má HTML niekoľko spoločných vlastností s XML. Oba používajú značky uzavreté v hranatých zátvorkách (< >), organizujú obsah do elementov s atribútmi a spoliehajú sa na stromové hierarchické štruktúry na vyjadrenie vzťahov medzi časťami dokumentu. Vďaka tejto štruktúrálnej podobnosti je jazyk HTML prístupný každému, kto už pozná jazyk XML, a naopak.

Kľúčové rozdiely sa však prejavujú v ich účele, dizajne a stupni prísnosti. Ako sme už videli, XML je navrhnutý ako flexibilný, univerzálny značkovací jazyk používaný na kódovanie a prenos údajov so silným dôrazom na štruktúrálne rigidnosť. Striktne vyžaduje korektné vnorenia prvkov, rozlišovanie veľkých a malých písmen značiek a explicitne uzavreté elementy. Naproti tomu HTML - najmä jeho moderná inkarnácia HTML5 - je prispôbený na zobrazovanie obsahu vo webových prehliadačoch a je navrhnutý s väčšou toleranciou voči syntaktickým chybám. Umožňuje vynechanie uzatváracích značiek a pri názvoch značiek nerozlišuje veľkosť písmen. Podstatným rozdielom je tiež fakt, že je tvorený obmedzeným súborom elementov s vopred definovanou sémantikou zameraných na vizuálnu a interaktívnu prezentáciu obsahu.³

Okrem toho, zatiaľ čo XML zvyčajne vyžaduje sprievodnú schému na validáciu štruktúry dokumentu, HTML dokumenty sa typicky interpretujú podľa štandardizovaného modelu vykresľovania zabudovaného v aplikáciách súhrne označovaných ako webové prehliadače.

³Je možné definovať aj vlastné HTML elementy, známe aj ako webové komponenty, ale proces ich definovania je technicky náročnejší, ako v prípade XML elementov, keďže si vyžaduje aspoň základnú schopnosť programovania v jazyku JavaScript. Pre bližšie oboznámenie sa s programovaním webových komponentov pozri (*Web Components - Web APIs / MDN*)

Vďaka tomu sa HTML ľahšie používa na vizuálnu prezentáciu obsahu, ale je menej prísny, pokiaľ ide o formálnu štruktúru dokumentu a vlastné slovníky.

Hoci existujú desiatky HTML elementov,⁴ niektoré z nich sa používajú častejšie vďaka ich základnej role pri rozvrhnutí obsahu, vytváraní odkazov, médiách a interakcii. Nižšie uvádzame dôležité komponenty bežného HTML dokumentu a opíšeme ich funkciu a typický obsah.

Prvým riadkom každého moderného HTML dokumentu je deklarácia typu dokumentu, ktorá prehliadaču oznamuje, aká verzia jazyka HTML sa používa. V HTML5 je táto deklarácia zámerne minimálna:

```
<!DOCTYPE html>
```

Hoci tento riadok nepredstavuje samotný HTML element, zohráva dôležitú úlohu, keďže zabezpečuje, aby prehliadač vykresľoval stránku v režime kompatibilnom so štandardmi na rozdiel od režimu “quirks” (režim spätnej kompatibility, ktorý sa snaží napodobniť staršie správanie prehliadačov). Táto jednoduchá deklarácia je dôležitou súčasťou vytvárania predvídateľného správania pri vykresľovaní naprieč rôznymi prehliadačmi.

Hneď za úvodnou nasleduje element `<html>`, ktorý je koreňovým elementom a obsahuje teda všetky ostatné elementy nachádzajúce sa v dokumente. Zvyčajne obsahuje atribút `@lang` indikujúci primárny jazyk dokumentu, čo je dôležité pre nástroje prístupnosti a optimalizáciu pre vyhľadávače.

```
<html lang="sk">
```

```
...
```

```
</html>
```

Prvým potomkom koreňového elementu je hlavička dokumentu reprezentovaná elementom `<head>`, ktorá obsahuje metadáta o dokumente, ktoré prehliadače užívateľovi bezprostredne nezobrazujú. Táto časť je kľúčová pre nastavenie kódovania znakov, prepojenie externých súborov štýlov a skriptov, definovanie názvu stránky a poskytovanie informácií pre prehliadače, vyhľadávače a sociálne platformy.

Typický obsah tohto elementu vyzerá nasledovne:

⁴Pre obsiahnejší prehľad pozri (*HTML Standard*) alebo (*HTML Reference - HTML / MDN*)

```

<head>

  <meta charset="UTF-8">

  <!-- Definuje kódovanie znakov, čím umožňuje podporu rôznych jazykov -->

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!-- Nastavuje responzívne rozloženie obsahu na mobilných zariadeniach -->

  <title>Web Stránka</title>

  <!-- Názov, ktorý sa zobrazuje na titulkovom riadku alebo karte prehliadača -->

  <link rel="stylesheet" href="style.css">

  <!-- Prepojenie na externý súbor CSS obsahujúci vizuálne štýly dokumentu -->

  <script src="script.js" defer></script>

  <!-- JavaScript súbor, ktorý sa spustí po načítaní dokumentu -->

</head>

```

Bezprostredne po hlavičke dokumentu nasleduje element `<body>`, ktorý obsahuje všetko, čo používatelia na webovej stránke skutočne vidia a s čím interagujú: text, obrázky, nadpisy, zoznamy, tabuľky, odkazy, formuláre, vložené médiá a ďalšie. Tu sa implementuje sémantická štruktúra obsahu pomocou rôznych značiek:

```

<body>

  <header>

    <h1>Digitálna zbierka prózy</h1>

  </header>

  <nav>

    <ul>

      <li><a href="#oprojekte">0 projekte</a></li>

      <li><a href="#zbierka">Zbierka</a></li>

      <li><a href="#kontakt">Kontakt</a></li>

    </ul>

  </nav>

  <main>

```

```
<section id="onas">
  <h2>0 projekte</h2>
  <p>...</p>
</section>
<section>
  <h2>Zbierka</h2>
  <p>...</p>
</section>
<section id="kontakty">
  <h2>Kontakty</h2>
  <p>...</p>
</section>
</main>
<footer>
  <p>&copy; 2025 Dispro</p>
</footer>
</body>
```

Obrázok 8 predstavuje, ako webové prehliadače zobrazia súbor, ktorý vznikne kombináciou vyššie uvedených HTML fragmentov v jednom dokumente.

Digitálna zbierka prózy

- [O projekte](#)
- [Zbierka](#)
- [Kontakt](#)

O projekte

...

Zbierka

...

Kontakty

...

© 2025 Dispro

Obrázok 8: Zobrazenie HTML súboru vo webovom prehliadači

CSS

CSS je jazyk používaný na ovládanie prezentácie, formátovania a rozvrhnutia HTML dokumentov. Zatiaľ čo jazyk HTML poskytuje štrukturálny rámec a sémantický význam webového obsahu, CSS je zodpovedné za to, ako tento obsah vyzerá - ako je štylizovaný, usporiadaný a zobrazený na rôznych zariadeniach a pri rôznych veľkostiach obrazovky. Toto jasné oddelenie štruktúry a štýlu je základným princípom moderného vývoja webových stránok.

CSS v podstate umožňuje vývojárom webových stránok špecifikovať pravidlá dizajnu - napríklad farby, písma, medzery, okraje, zarovnanie a responzivitu - ktoré sa vzťahujú na HTML elementy. Takto zabezpečuje konzistentný a upravený vzhľad celej webovej lokality, a to aj pri zmene alebo rozšírení obsahu. CSS umožňuje všetko od základných štýlov, ako je tučné alebo farebné písmo, až po pokročilé techniky rozvrhnutia, ako sú viacstĺpcové usporiadania obsahu a responzívne mriežky.

Jednou z kľúčových princípov jazyka CSS je “kaskáda”, ktorá označuje spôsob, akým sa pravidlá štýlu uplatňujú v kontextoch, keď sa viacero pravidiel zameriava na ten istý element. Tento mechanizmus zohľadňuje faktory, ako je špecifickosť, poradie zdrojov a dôležitosť (napríklad pravidlá označené ako **!important**). Umožňuje to zachovať jemnú kontrolu nad vzhľadom jednotlivých elementov a zároveň udržiavať kód organizovaný a udržiavateľný.

CSS pravidlá možno definovať tromi hlavnými spôsobmi. Ako inline štýly v podobe zoznamu uvedeného v hodnote atribútu `@style` elementu, na ktoré sa dané pravidlá aplikujú:

```
<h1 style="color:red; ..."></h1>
```

Ďalším spôsobom je zapísanie pravidiel v bloku `<style>` nachádzajúcom sa v hlavičke (`<head>`) HTML dokumentu:

```
<head>
...
<style>
  h1 {
    color: red;
```

```
    }  
    ...  
  </style>  
  ...  
</head>
```

A napokon môžeme definovať CSS pravidlá v externom súbore, ktorý prepojíme s HTML dokumentom prostredníctvom elementu `<link rel="stylesheet" href="style.css">`⁵ uvedenom v hlavičke (`<head>`) dokumentu.

Každé CSS pravidlo sa zameriava na konkrétne prvky a uplatňuje na ne súbor štylistických pokynov. Syntax pravidiel je jednoduchá a zároveň expresívna, navrhnutá tak, aby bola čitateľná pre človeka a zároveň poskytovala efektívnu kontrolu nad prezentáciou dokumentu. Pravidlo sa skladá z dvoch hlavných častí: *selektora* a *deklaračného bloku*.

CSS selektor určuje, na ktoré elementy sa pravidlo vzťahuje. Môže odkazovať na prvky podľa názvu (napr. `p` pre značky odsekov), podľa triedy (napr. `.highlight`), podľa ID (napr. `#hlavny-napdis`) alebo podľa zložitejších vzorov, ako sú vnorené alebo atribútové selektory.

Deklaračný blok je uzavretý v kučeravých zátvorkách `{}` a obsahuje jednu alebo viac deklarácií, z ktorých každá špecifikuje vlastnosť a hodnotu, oddelené dvojbodkou a ukončené stredníkom. Tieto deklarácie definujú, aký vizuálny štýl sa má použiť na selektorom vymedzené elementy.

Pre ilustráciu si vezmeme príklad, kde pre všetky paragrafy v dokumente (vymedzené selektorom `p`) nastavíme modrú farbu písma, 16 pixelovú veľkosť fontu a veľkosť riadkovania na 1½ násobok veľkosti fontu:

```
p {  
  color: blue;  
  font-size: 16px;  
  line-height: 1.5;  
}
```

⁵Hodnota atribútu `@href` predstavuje cestu k súboru obsahujúcom CSS pravidlá.

Okrem selektorov, ktoré identifikujú HTML elementy, na ktoré sa majú uplatniť štýly uvedené v deklaračnom bloku skrz ich typ, existujú aj ďalšie formy selektorov:

Triedne selektory cielia na elementy s určitými hodnotami atribútu `@class`. Všetky takéto selektory majú `.` prefix:⁶

```
.note {
  background-color: yellow;
}
```

ID selektory cielia na elementy s určitou hodnotou atribútu `@id`. Všetky takéto selektory majú `#` prefix:⁷

```
#main-header {
  font-size: 24px;
}
```

Univerzálny selektor `*` cieli na všetky elementy nachádzajúce sa v dokumente:⁸

```
* {
  color: red;
}
```

Druh			
selektoru	Syntax	Príklad	Efekt
Typ	meno elementu	<code>p</code>	Všetky paragrafy
Trieda	<code>.názov-triedy</code>	<code>.note</code>	Všetky elementy s triedou <code>note</code>
ID	<code>#hodnota-id</code>	<code>#main</code>	Element s id “main”
Potomok	<code>A B</code>	<code>article p</code>	Všetky paragrafy v článkoch
Priamy potomok	<code>A > B</code>	<code>ul > li</code>	Prvoúrovňové položky <code></code> zoznamov <code></code>
Susediaci súrodenec	<code>A + B</code>	<code>h2 + p</code>	Všetky paragrafy za nadpismi

⁶Uvedený kód nastaví pozadie všetkých elementov ako `<div class="note">...</div>` na žltú farbu.

⁷Uvedený kód nastaví veľkosť fontu elementu s identifikátorom nastaveným na “main-header” (aplikovateľné napríklad pre element `<h1 id="main-header">...</h1>`) na 24 pixelov.

⁸Uvedený kód nastaví farbu všetkých elementov v dokumente na červenú.

Druh			
selektoru	Syntax	Príklad	Efekt
Atribút	[atr.=hod.]	<code>input [type="text"]</code>	elementy typu "text"
Zoskupenia	A, B	<code>h1, h2</code>	Nadpisy prvej a druhej úrovne
Pseudo triedy	<code>:názov-pseudo-triedy</code>	<code>a:hover</code>	

TEI Publisher

Po preskúmaní niektorých webových technológií sa teraz zameriame na špecializovanú platformu určenú pre vedcov pracujúcich s bohato kódovanými textovými údajmi: *TEI Publisher*. Ovládanie základných technológií nám síce poskytuje veľkú flexibilitu a kontrolu nad výsledným produktom, ale ich integrácia so štruktúrne komplexnými TEI dokumentmi si často vyžaduje enormné úsilie. TEI Publisher preklenuje túto medzeru tým, že ponúka rámec, ktorý prekladá TEI XML priamo do dynamických, štylizovaných a prehľadateľných webových rozhraní. Využíva výrazovú silu XML spolu s webovými štandardmi na vytváranie prístupných vedeckých publikácií bez rozsiahleho manuálneho programovania. V nasledujúcej časti preskúmame, ako TEI Publisher funguje, na akých technológiách stavia a ako uľahčuje webovú prezentáciu digitálnych textových korpusov.

Technológie

TEI Publisher je postavený na eXist-db, natívnej XML databáze, ktorá umožňuje dopytovanie, transformáciu a správu údajov prispôbenú XML dokumentom. Poskytuje podporu XQuery, indexovanie, RESTful API a natívne úložisko XML, ktoré sú nevyhnutné na efektívne sprostredkovanie a manipuláciu TEI dokumentov.

Na vyhľadávanie a manipuláciu s XML TEI Publisher vo veľkej miere využíva XQuery a XPath, ktoré umožňujú presný výber, transformáciu a filtrovanie TEI elementov. Tieto jazyky tvoria logickú kostru dynamického generovania obsahu a podporujú všetko od fazetového vyhľadávania po krížové odkazy a agregáciu údajov.

TEI Publisher používa XSLT na konverziu TEI dokumentov do HTML formátu, čím umožňuje ich zobrazenie vo webových prehliadačoch. Tieto transformácie sprístupňujú

užívateľom bohaté možnosti prispôsobiteľnosti, napríklad zobrazovanie XML štruktúr do formátov čitateľných pre človeka, formátovanie metadát a generovanie navigačných pomôcok, ako sú napríklad obsahy.

Konečná prezentačná vrstva sa spracúva pomocou štandardných webových technológií, ktorým sme sa venovali v predchádzajúcich častiach.

TEI Publisher tiež využíva aj niekoľko komponentov založených na jazyku Java z ekosystému eXist-db vrátane knižníc na špecifické spracovanie TEI, extrakciu metadát a validáciu schém.

Tieto technológie spoločne umožňujú platforme slúžiť ako most medzi štruktúrovanými vedeckými údajmi a prístupnou, používateľsky prívetivou webovou publikáciou. Vďaka modulárnemu dizajnu a spoliehaniu sa na nástroje založené na štandardoch je robustný a prispôsobivý pre rôzne projekty digitálnych humanitných vied, od dokumentov s jedným vydaním až po rozsiahle korpusy.

Ako TEI Publisher funguje

Ako sme spomenuli vyššie TEI Publisher ukladá dokumenty do databázy eXist-db, ktorá nielen uchováva TEI súbory, ale umožňuje aj ich indexovanie a vyhľadávanie. Texty tak zostanú štruktúrované a je vďaka tomu možné vyhľadávať v ich obsahu a dynamicky ich poskytovať používateľom bez sploštenia alebo konverzie zdrojového XML do neštruktúrovaných formátov.

Pre sprístupnenie TEI dokumentov na webe transformuje *Publisher* XML do HTML prostredníctvom viacúrovňového procesu, ktorý kombinuje deklaratívne transformačné jazyky so šablónami ovládanými konfiguráciou. Tento proces využíva štandardné XML technológie - najmä XSLT (Extensible Stylesheet Language Transformations) a XQuery - ktoré spoločne interpretujú sémantiku kódovania TEI a vykresľujú ju do štruktúrovaných, štylizovaných a interaktívnych webových stránok.

Úloha XSLT: od TEI k HTML

Základom transformácie je jazyk XSLT, ktorý je špeciálne navrhnutý na konverziu dokumentov XML do iných formátov. TEI Publisher sa dodáva s robustnou sadou XSLT šablón

prispôsobených pre TEI. Tie definujú, ako majú byť jednotlivé prvky TEI - ako `<div>`, `<p>`, `<pb>`, `<note>`, `<hi>` alebo `<persName>` - reprezentované v jazyku HTML. Napríklad:

- `<div type="chapter">` môže byť transformovaný do HTML elementu `<section>`.
- Z `<note>` sa môže stať poznámka pod čiarou alebo tooltip, v závislosti od konfigurácie.
- `<pb/>` (prerušenie stránky) môže byť vykreslený ako vizuálna indikátor stránky alebo ako prepojenie na faksimile.

Tieto transformácie sú rekurzívne a citlivé na kontext, čo znamená, že výstupná štruktúra HTML rešpektuje vnorenie aj sémantiku zdrojových XML súborov.

XQuery ako kontrolér a smerovač

XQuery zohráva úlohu orchestrácie. V eXist-db databáze vyhľadáva relevantné dokumenty a fragmenty TEI a určuje, ako ich zostaviť alebo odoslať na transformáciu. Napríklad, keď používateľ prejde na konkrétny dokument alebo časť edície, XQuery skripty načítajú príslušný(-é) súbor(-y), preženu ich transformačným kanálom XSLT a vložia výsledné HTML do webového rozhrania.

Okrem toho sa XQuery môže použiť na extrakciu metadát na filtrovanie, generovanie obsahových náhľadov a spracovanie funkcií vyhľadávania priamym dopytovaním sa na metadáta TEI alebo prvky obsahu.

Šablóny a konfigurácia

TEI Publisher abstrahuje veľkú časť zložitosti XSLT a XQuery procesov prostredníctvom systému konfiguračných súborov a šablón napísaných v jazyku XML. Tie definujú veci ako:

- Ktoré prvky TEI sa majú zobrazíť alebo ignorovať
- Ako štylizovať určité značky
- Ktoré polia metadát sa zobrazujú vo vyhľadávacích filtroch
- Aké zobrazenia (napr. plný text, faksimile, súhrn metadát) sú k dispozícii

Táto modularita umožňuje projektovým tímom prispôbiť TEI Publisher rôznym vedeckým požiadavkám bez toho, aby museli sami písať nízkoúrovňový transformačný kód.

Vykresľovanie a interakcia

Po transformácii do jazyka HTML sa obsah naštýluje pomocou jazyka CSS a interaktívne aspekty sa vytvoria pomocou jazyka JavaScript. Tieto frontendové nástroje zvládajú dynamické funkcie, ako napr:

- Skladacie sekcie
- Vyskakovacie okná pre anotácie alebo odkazy
- Rozhrania pre fazetové vyhľadávanie podľa
- Porovnávacie zobrazenia (napr. text vs. obrázok)

Dôležité je, že sa to všetko vykonáva bez zmeny základného formátu TEI XML, ktorý zostáva zachovaný a je možné v ňom vyhľadávať v jeho pôvodnej podobe.

TEI Publisher teda funguje ako sofistikovaný transformačný a publikačný kanál. Mapuje sémanticky bohatý, štruktúrovaný svet TEI XML na prístupné, vizuálne a interaktívne médium HTML. Využíva na to XSLT na transformáciu, XQuery na riadiacu logiku a vrstvu šablón a konfigurácií, ktoré umožňujú humanistom vytvárať digitálne edície bez toho, aby potrebovali hlboké skúsenosti s vývojom webu.

Bibliografia

“Action CA16204.” In *COST*. n.d. Accessed June 10, 2025. <https://www.cost.eu/actions/CA16204>.

DeRose, S. J. *The SGML FAQ Book: Understanding the Foundation of HTML and XML*. 1997th edition. Kluwer Academic Publishers, 1997.

Extensible Markup Language (XML) 1.0 (Fifth Edition). n.d. Accessed April 15, 2025. <https://www.w3.org/TR/REC-xml/>.

Fennell, Philip. “Extremes of XML.” *XML London 2013 Conference Proceedings*, XML London, June 2013, 80–86. <https://doi.org/10.14337/XMLLondon13.Fennell01>.

Getting Started with P5 ODDs. n.d. Accessed June 3, 2025. <https://tei-c.org/guidelines/customization/getting-started-with-p5-odds/>.

“Git.” In *Wikipedia*. 2025. <https://en.wikipedia.org/w/index.php?title=Git&oldid=1288616179>.

Gulbrandsen, David, Kynn Bartlett, Earl Bingham, Alexander Kachur, Kenrick Rawlings, and Andrew H. Watt. *Special Edition Using XML, 2nd Edition*. 2nd ed. Que., 2002. https://www.informit.com/store/special-edition-using-xml-9780789727480?w_ptgrevartcl=XML+Building+Blocks%3a+Elements+and+Attributes_27865.

HTML Reference - HTML / MDN. 2025. <https://developer.mozilla.org/en-US/docs/Web/HTML/Reference>.

Bibliografia

HTML Standard. n.d. Accessed June 14, 2025. <https://html.spec.whatwg.org/multipage/>.

“ISO/IEC 8859-1.” In *Wikipedia*. 2025. https://en.wikipedia.org/w/index.php?title=ISO/IEC_8859-1&oldid=1299638266.

Nielsen, Jakob. *Multimedia and Hypertext: The Internet and Beyond*. Morgan Kaufmann, 1995.

RELAX NG Home Page. n.d. Accessed May 31, 2025. <https://relaxng.org/>.

The TEI Guidelines. n.d. Accessed May 31, 2025. <https://tei-c.org/release/doc/tei-p5-doc/en/html/index.html>.

Unicode Standard. n.d. Accessed April 15, 2025. <https://www.unicode.org/standard/standard.html>.

Vannevar Bush. *As We May Think*. 1945. <http://archive.org/details/as-we-may-think>.

Web Components - Web APIs / MDN. 2025. https://developer.mozilla.org/en-US/docs/Web/API/Web_components.

“What Is XML (Extensible Markup Language)?” In *WhatIs*. n.d. Accessed April 16, 2025. <https://www.techtarget.com/whatis/definition/XML-Extensible-Markup-Language>.

XSL Transformations (XSLT) Version 2.0 (Second Edition). n.d. Accessed April 22, 2025. <https://www.w3.org/TR/xslt20/>.

XSL Transformations (XSLT) Version 2.0 (Second Edition). n.d. Accessed July 25, 2025. <https://www.w3.org/TR/xslt20/>.

Budovanie digitálnych zbierok 1. (Technologické minimum)

Marek Debnár - Marek Vician

Vydavateľ: Univerzita Konštantína Filozofa v Nitre

Prvé vydanie, Nitra 2025

Počet strán: 70

ISBN 978-80-558-2289-1

