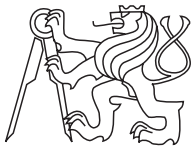
České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství
Obor: Aplikace softwarového inženýrství



Porovnání účinnosti komprese dat ve formátech XML a JSON

Comparison of the effectiveness of data compression in XML and JSON format

DIPLOMOVÁ PRÁCE

Vypracoval: Bc. Tomáš Smola
Vedoucí práce: Ing. Tomáš Liška, Ph.D.
Rok: 2015

Před svázáním místo téhle stránky vložte zadání práce s podpisem děkana (bude to jediný oboustranný list ve Vaší práci) !!!!

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne .....

.....
Bc. Tomáš Smola

Poděkování

Děkuji Ing. Tomáši Liškovi, Ph.D. za vedení mé diplomové práce a za podnětné návrhy, které ho obohatily.

Bc. Tomáš Smola

Název práce:
Porovnání účinnosti komprese dat ve formátech XML a JSON

Autor: Bc. Tomáš Smola

Obor: Aplikace softwarového inženýrství
Druh práce: Diplomová práce

Vedoucí práce: Ing. Tomáš Liška, Ph.D.
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze
Konzultant: —

Abstrakt: Abstrakt

Klíčová slova: Klíčová slova

Title:
Comparison of the effectiveness of data compression in XML and JSON format

Author: Bc. Tomáš Smola

Abstract: Abstract

Key words: Key words

Obsah

Úvod

Závěrečnou diplomovou práci ke studijnímu oboru Aplikace softwarového inženýrství s názvem Porovnání účinnosti komprese dat ve formátu XML a JSON jsem si vybral z důvodu aktuálnosti – XML a JSON jsou v současnosti jedny z nejpoužívanější textových datových formátů – a také proto že toto téma velmi dobře propojuje teoretické znalosti získané při studiu s praktickými zkušenostmi v oboru softwarového inženýrství.

Dle výzkumu International Data Corporation (IDC) The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things [2] bylo pouze v roce 2014 vytvořeno a zkonzumovýno 2837 EB (exabytů) dat. Ze závěrů vyplývá, že se toto číslo každé dva roky zdvojnásobí, tedy v roce 2020 to bude již přibližně 40000 EB. Takové množství dat klade enormní požadavky na přenosové kanály a datová úložiště. Jako příklad mohou sloužit miliony shlédnutí oblíbených videí na serveru youtube.com. Pouze jedna sekunda videa v nekomprimovaném formátu CCIR 601 zabere více než 20 MB, tímto způsobem by zmiňovaná služba nemohla fungovat.

Vzhledem k tomu, že jsou technologie omezeny současnými možnostmi, znalostmi a také fyzikálními limity, je nutné hledat řešení jinde než v jejich zlepšování. Zde přichází na řadu komprese dat jako účinná metoda snížení velikosti objemu přenášených a ukládaných dat. Ve své práci bych čtenáře rád seznámil se základními principy komprese a vybranými kompresními algoritmy. Hlavním cílem je ale zodpovědět otázku, zda je možné dosáhnout dalších úspor volbou XML nebo JSON formátu a vhodného algoritmu, který využije znalosti struktury datového formátu.

Kapitola 1

Obecné seznámení s formáty XML a JSON

V této kapitole seznámím čtenáře se značkovacím jazykem XML a následně s JSONem, formátem pro výměnu dat. Mým cílem je popsat základní charakteristiky a syntaxi obou formátů tak, abych byl já, a následně i čtenář, schopen pochopit v kapitole 4 principy a výhody vybraných algoritmů využívajících znalosti struktury datových souborů.

1.1 XML

Na základě značkovacího jazyka SGML (Standard Generalized Markup Language), jehož obecnost činí úplnou implementaci velmi náročnou, vznikl vybráním nejpoužívanějších možností nový značkovací jazyk XML (eXtensible Markup Language), je tedy podmnožinou jazyka SGML. XML je obecný a otevřený, jeho vývoj a standardizaci realizovalo konsorcium W3C (World Wide Web Consorcium) [3]. XML umožňuje snadné vytváření konkrétních značkovacích jazyků pro popis dokumentů a dat ve standardizované, textově orientované podobě.

1.1.1 Charakteristika

V podstatě jde o textový dokument, jež je tvořen posloupností Unicode[1] znaků, ve kterém se rozlišují dva základní prvky: elementy (neboli značky) a obsah. Strukturu zapisovaných dat je možné vystihnout vzájemnému vnořování jednotlivých značek, které se ale nesmí křížit.

XML je pro člověka čitelné a spousta dnešních programů (například internetových prohlížečů) podporou zobrazení a vhodným formátováním tuto čitelnost ještě zvyšuje.

Jazyk XML se využívá hlavně pro publikování dokumentů, při výměně dat mezi různými systémy v prostředí internetu, jako univerzální úložiště dat či jako konfigurační soubor. Obecně lze říci, že je vhodný pro strukturovaná data. Do dokumentu můžeme vložit libovolná data, jejichž význam není bez použití schématu dokumentu zřejmý. Abychom mohli definovat sadu elementů (viz následující část 1.1.2), případně v nich i kontrolovat

[1] Unicode je standard pro konzistentní kódování, reprezentaci a manipulaci znaků většiny světových abeced.

typ dat, definujeme schéma dokumentu, které je vloženo buď přímo, nebo formou odkazu na definiční dokument.

1.1.2 Syntaktická analýza

Jak bylo řečeno již dříve, základními kameny XML jsou elementy a jejich obsah. Při práci s XML je nutné mít na paměti, že je na dodržení syntaxe kladen velmi velký důraz. Při dodržení správného způsobu zápisu a pravidel, která budou popsána níže, lze dokument považovat za tzv. well-formed XML [3].

Element

Základním prvkem každého XML dokumentu je element, který je vyznačen pomocí takzvaných tagů[2], mezi které může být vložen obsah. Počáteční i ukončující tag je dle definice [3] složen z dvojice znamének `<` (menší než) a `>` (větší než), mezi kterými je zapsán název tagu a volitelně i atributy. Ukončovací tag má navíc před svým názvem znak `/` (lomeno). Při správné aplikaci pravidel může vypadat element například následujícím způsobem:

```
<název_elementu název_atributu="hodnota atributu"></název_elementu>.
```

V případě, že element neobsahuje žádný obsah, lze ho zkráceně zapsat jako tzv. prázdný element:

```
<název_prázdného_elementu />.
```

V případě nedodržení správné syntaxe může nastat problém při rozpoznávání zapsaných dat, což může mít za následek nekompatibilitu mezi různými systémy při výměně dat.

Atribut

Počáteční tag elementu může obsahovat atributy upřesňující jeho význam. Atribut je vždy složen ze svého názvu a hodnoty, které jsou odděleny znakem `=` (rovná se). Hodnota je navíc zapsána mezi dvojici znaků `"` (uvozovky) nebo `'` (apostrof), přičemž hodnota může obsahovat jeden z těchto znaků tak, že se syntakticky nekříží. Následuje příklad atributu, jehož hodnota obsahuje znak `'`:

```
název_atributu="hodnota atributu osahující znak ' (apostrof)".
```

Obsah

Vše, co není tagem, je v dokumentu považováno za obsah. Kromě obyčejného textu mohou být obsahem další vnořené elementy, komentáře, instrukce pro zpracování, reference a další. Vzhledem k tomu, že určité znaky mají v syntaxi XML speciální význam (např. `<`, `>`), využívají se pro jejich zápis znakové entity[3]. Úplný výčet toho, co může XML dokument obsahovat, je včetně pravidel definován v [3].

[2] Tag definuje formu části textu.

[3] Pomocí znakových entit (sekvence znaků) lze zapsat znaky, které neobsahuje zvolená znaková sada, nebo mají v použitém kontextu speciální význam.

1.1.3 Vzorový příklad

Na následujících řádcích je zapsán XML element typu `Person`, oblíbená postavička Homer Simpson ze seriálu The Simpsons, který kromě jiného obsahuje vnořený element typu `Relatives`, který zde slouží jako kontejner s Homerovými příbuznými.

```
<Person>
 <FirstName>Homer</FirstName>
 <LastName>Simpson</LastName>
 <Relatives>
  <Relative>Grandpa</Relative>
  <Relative>Marge</Relative>
  <Relative>Bart</Relative>
  <Relative>Lisa</Relative>
  <Relative>Maggie</Relative>
 </Relatives>
</Person>
```

1.2 JSON

JSON neboli JavaScript Object Notation je odlehčený způsob zápisu (formátování) dat. Navzdory svému názvu jde o datově orientovaný textový formát nezávislý na počítačové platformě a čitelný pro člověka. Vznikl jako alternativa ke XML v oblasti výměny dat mezi systéme bez ohledu na použité technologie.

1.2.1 Charakteristika

Jak již název napovídá, je JSON velice úzce spojen s programovacím jazykem JavaScript – je na jeho syntaxi je založen. Hlavními prvky JSONu jsou dvě univerzální datové struktury: kolekce dvojic klíč/hodnota s unikátními klíči (tzv. associative array) a seřazený seznam hodnot (tzv. array), které podporují v nějaké formě asi všechny známé moderní programovací jazyky.

Silnou stránkou JSONu je rychlost zpracování JavaScriptem, bohužel právě přímé zpracování je zároveň potenciální hrozbou, neboť může být zneužito útočníkem k vykonání nějakého škodlivého kódu.

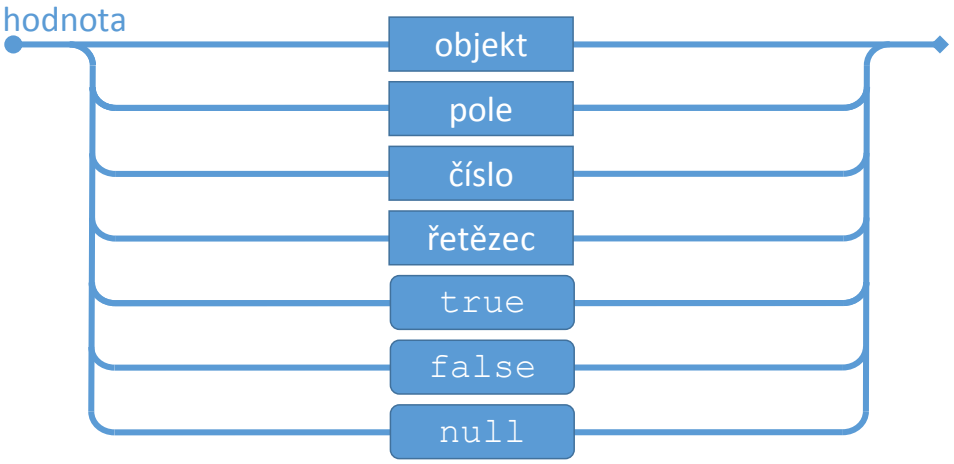
Orientace na data a zvolená syntaktická struktura, umožňují přehledný zápis datových objektů. Protože co jiného než než popis atributů (asociativní pole) a výčet hodnot (pole) datové objekty obsahují?

1.2.2 Syntaktická analýza

Dle definice [1] je JSON posloupností tokenů (viz 1.2.2) tvořených z Unicode znaků. Sada tokenů obsahuje šest strukturálních tokenů: `[` (levá hranatá závorka), { (levá složená závorka), `]` (pravá hranatá závorka), } (pravá složená závorka), `:` (dvojtečka) a `,` (čárka); dále obsahuje znakové řetězce, čísla a tři doslovné tokeny: `true`, `false` a `null`.

Hodnoty

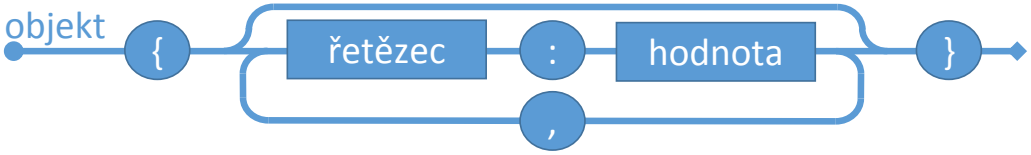
Za hodnotu je v JSONu považován objekt, pole, číslo, řetězec, `true`, `false`, nebo `null`.



Obrázek 1.1: Struktura hodnoty

Objekty

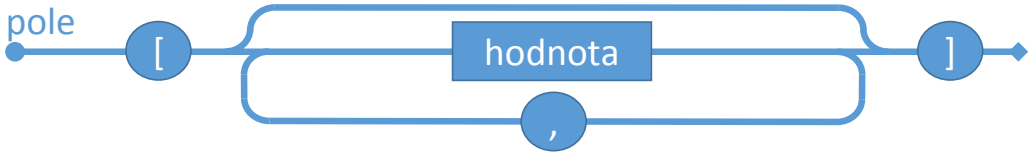
Objekt je reprezentován dvojicí složených závorek, uvnitř kterých je žádná nebo více dvojic klíč/hodnota, přičemž klíč je řetězec. Klíč a hodnota jsou odděleny dvojtečkou a jednotlivé dvojice odděluje čárka.



Obrázek 1.2: Struktura objektu

Pole

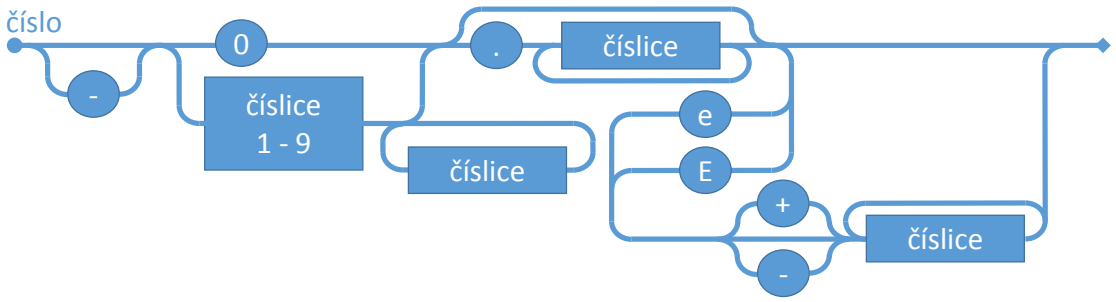
Pole je složeno z dvojice hranatých závorek, mezi kterými může být nula nebo více seřazených hodnot, které jsou odděleny čárkou.



Obrázek 1.3: Struktura pole

Čísla

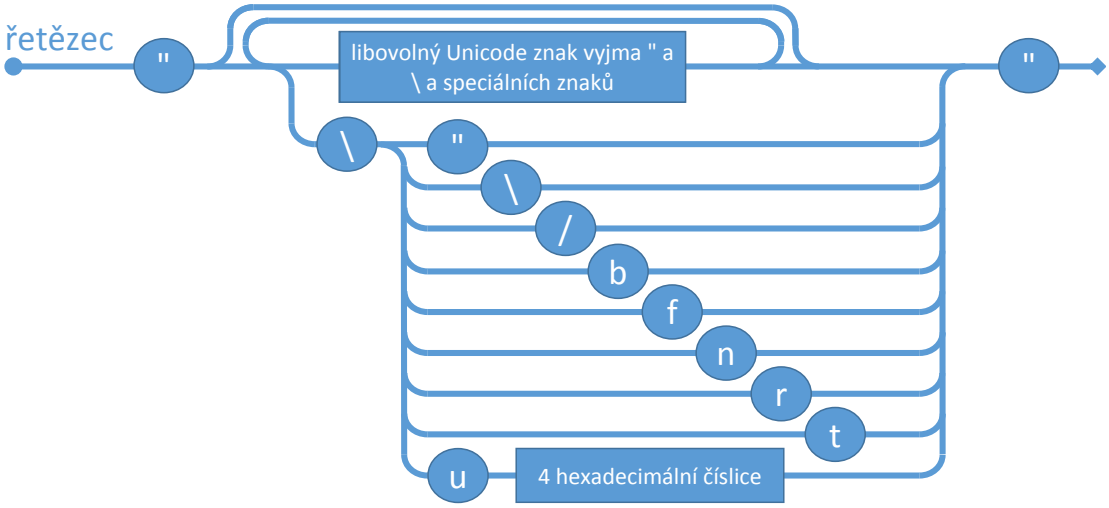
Čísla jsou v desítkové soustavě (tedy číslice $0-9$), záporná čísla jsou uvozena znaménkem `-` (mínus), desetinná část je oddělena znaménkem `.` (tečka). Je možný i takzvaný vědecký zápis čísel s použitím symbolů `e` (malé e) nebo `E` (velké e) a volitelně lze použít u exponentu znaménka `+` (plus) nebo `-` (mínus).



Obrázek 1.4: Struktura čísla

Řetězce

Řetězec je posloupnost Unicode znaků uvozená a zakončená znakem `"` (uvozovky). Mezi uvozovky mohou být zapsány všechny znaky kromě speciálních znaků, které jsou uvozeny tzv. únikovým znakem \ (zpětné lomítko), těmito znaky jsou například `"`, \, `t` (znak tabulátoru), `n` (znak posunu kurzoru na nový řádek) a `r` (znak posunu kurzoru na začátek řádku, známý jako návrat vozíku) a další. Kompletní výčet je možné vidět na obrázku 1.5 nebo v [1]. Navíc je možné každý znak zapsat pomocí kombinace `\u` a čtyřmístného hexadecimálního čísla odpovídajícího Unicode kódu požadovaného znaku. Řetězec obsahující pouze zpětné lomítko můžeme tedy zapsat následujícími způsoby: `"\\"`, `"\u005c"` nebo `"\u005C"` (u hexadecimálních čísel `A-F` nezáleží na velikosti).



Obrázek 1.5: Struktura řetězce

1.2.3 Vzorový příklad

Stejně jako ve vzorovém příkladě 1.1.3 ke XML, je zde popsána postavička Homera Simpsona včetně příbuzných, tentokrát ale v notaci JSON. Jde o objekt složený ze tří atributů, první dva mají hodnoty typu řetězec, k poslednímu `relatives` ale přísluší hodnota typu pole, ve kterém je vloženo pět příbuzných – hodnot typu řetězec.

```
{
 "firstName": "Homer",
 "lastName": "Simpson",
 "relatives": [
  "Grandpa",
  "Marge",
  "Bart",
  "Lisa",
  "Maggie"
 ]
}
```

1.3 Vzájemné porovnání XML a JSON

Je XML obecně lepší než JSON? Co vlastně dělá jeden formát lepší než jiný? Na toto bylo vedeno již nespočet diskusí. Argumentace obou stran diskuse je obvykle podložena jak teoretickou znalostí formátů, tak i praktickým použitím v reálných projektech. I přes to si myslím, že žádná diskuse nepřinesla jednoznačnou odpověď na výše položenou otázku, což samozřejmě pro některé konkrétní případy použití neplatí a volba jednoho z formátů je nutná nebo alespoň výhodnější. Ze svého zkoumání této problematiky si ale odnáším jeden podstatný závěr, který bych chtěl čtenáři zdůraznit: Vždy záleží na konkrétní situaci. Před samotným rozhodnutím, který z formátů využít, je vhodné zvážit důležité požadavky a do těchto úvah zahrnout klidně i další z existujících datových formátů. Zároveň lze ale říci, že dokud splňuje vybraný formát požadavky, tak ho lze považovat za dostatečný

Podíváme-li se znovu na vzorové příklady zápisu dat nebo položíme-li si vedle sebe stejná data zapsaná do obou formátů (viz příloha A.1), můžeme lehce vidět vzájemnou podobnost. Ve své práci proto vycházím z předpokladu, že jsou oba formáty z hlediska toho, jaká data lze do nich zapsat, ekvivalentní. Respektive budu pro další porovnání používat pouze takový druh dat, který bude tento předpoklad splňovat.

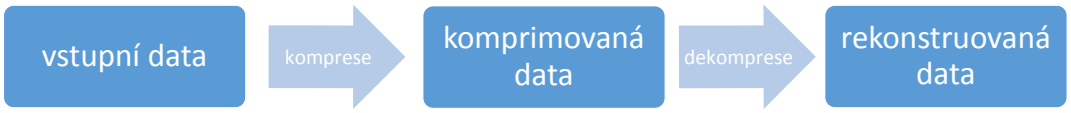
Kapitola 2

Komprese dat

Komprese nebo také komprimace dat je taková transformace dat, která má za cíl úsporu zdrojů při ukládání nebo archivaci a nebo snížení datového toku při přenosu, to vše při současném zachování informace obsažené v datech. Jinými slovy jde o redukci velikosti datových souborů, jehož následkem je úspora paměťových či přenosových kapacit. Postup, při kterém z komprimovaných dat rekonstruujeme data originální, se nazývá dekomprese.

2.1 Princip komprese dat

Data velmi často obsahují tzv. redundantní[1] informaci, toho právě využívá komprese – data jsou zpracována tak, aby byla redundance minimalizována. Jak lze vidět na obrázku 2.1, je na vstupní data použita operace komprese. Operací dekomprese dostaneme poté data rekonstruovaná – v závislosti na použité kompresní metodě, respektive na požadavcích získáme buď data přesně odpovídající původním a nebo pouze částečná. Z tohoto hlediska rozlišujeme dva typy kompresních metod: ztrátové a bezeztrátové.



Obrázek 2.1: Princip komprese

2.2 Typy kompresních metod

Jak název napovídá, při ztrátové kompresi ztratíme část informace obsaženou v původních datech, respektive jsou původní data pouze aproximována. Toto nám nemusí vadit například u obrázků, zvuku a videí, kde je využito nedokonalosti lidských smyslů. Lidské ucho nedokáže například slyšet velmi vysoké frekvence. Má smysl v datech určených k poslechu zachovávat informaci, kterou nemůže člověk slyšet? Častá odpověď je „ne“. Tohoto principu využívá mnoho kompresních metod, například známý zvukový formát MP3. Odstraněním „nepotřebné“ informace z dat je dosaženo ještě větší redukce objemu.

[1] Redundance znamená informační nadbytek, např. vícenásobný výskyt slov v textu.

Naopak v případě bezeztrátových metod je při kompresi zachována veškerá informace a při dekompresi jsou rekonstruována původní data. Těchto metod se využívá převážně tam, kde není možné původní data jakkoliv pozměnit. Například data ve formátech XML a JSON, kterým se věnuji v této práci, si nemůžeme dovolit pozměnit (přestanou mít původní význam), nebo dokonce ztratit.

2.3 Charakteristika komprese

Kompresní algoritmy lze hodnotit z mnoha různých úhlů pohledu. Můžeme měřit složitost algoritmu, rychlost, jakou data komprimována a dekomprimována (to může být ovlivněno výkonem stroje, na kterém algoritmus běží), jak moc odpovídají rekonstruovaná data původním atd. Jednou z nejčastějších charakteristik je, logicky ze smyslu komprese vyplývající, tzv. kompresní poměr, který vyjadřuje velikost komprimovaných dat vůči původním, lze ho zapsat následujícím vztahem:

$$\texttt{kompresní poměr} = \frac{\texttt{délka původních dat}}{\texttt{délka komprimovaných dat}}. \tag{2.1}$$

Další sledovanou charakteristikou je tzv. úspora místa, která je vyjádřena jako:

$$\texttt{úspora místa} = 1 - \texttt{kompresní poměr}^{-1}. \tag{2.2}$$

Mějme například 2D obrázek o velikosti 256×256 pixelů, který zabírá 65536 bytů. Obrázek zkomprimujeme a zabírá-li komprimovaná verze 16384 bytů, můžeme říct, že kompresní poměr je $4 : 1$ a úspora místa 75 %.

2.4 Míra informace v datech

Kapitola 3

Popis existujících kompresních algoritmů

Kapitola 4

Přehled existujících implementací kompresních algoritmů pro efektivní uchovávání dat ve formátu XML a JSON

Kapitola 5

Vlastní implementace vybraných kompresních algoritmů

Kapitola 6

Porovnání účinnosti komprese dat ve formátu XML a JSON

Závěr

Seznam použitých zdrojů

[1] Standard ECMA-404. *The JSON Data Interchange Format.* Geneva: Ecma International, 2013, [online], [cit. 28. října 2014]. Dostupné na: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>.

[2] The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. INTERNATIONAL DATA CORPORATION. *EMC* [online]. 2014 [cit. 2015-02-25]. Dostupné z: <http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm>.

[3] W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition).* 26. listopadu 2008, [online], [cit. 26. listopadu 2014]. Dostupné na: <http://www.w3.org/TR/2008/REC-xml-20081126/>.

Přílohy

Příloha A

Název/obsah přílohy

A.1 Porovnání XML a JSON

XML

```
<widget>
 <debug>on</debug>
 <window
  title="Sample Widget">
  <name>main_window</name>
  <width>500</width>
  <height>500</height>
 </window>
 <image
  src="Images/Sun.png"
  name="sun1">
  <hOffset>250</hOffset>
  <vOffset>250</vOffset>
  <alignment>center</alignment>
 </image>
 <text
  data="Click Here"
  size="36"
  style="bold">
  <name>text1</name>
  <hOffset>250</hOffset>
  <vOffset>100</vOffset>
  <alignment>center</alignment>
  <onMouseUp>sun1.opacity =
   (sun1.opacity / 100) * 90;
  </onMouseUp>
 </text>
</widget>
```

JSON

```
{ "widget": {
 "debug": "on",
 "window": {
  "title": "Sample Widget",
  "name": "main_window",
  "width": 500,
  "height": 500
 },
 "image": {
  "src": "Images/Sun.png",
  "name": "sun1",
  "hOffset": 250,
  "vOffset": 250,
  "alignment": "center"
 },
 "text": {
  "data": "Click Here",
  "size": 36,
  "style": "bold",
  "name": "text1",
  "hOffset": 250,
  "vOffset": 100,
  "alignment": "center",
  "onMouseUp": "sun1.opacity =
   (sun1.opacity / 100) * 90;"

 }
}}
```