

**EKONOMICKÁ UNIVERZITA V BRATISLAVE**

**FAKULTA HOSPODÁRSKEJ INFORMATIKY**

Evidenčné číslo: 103004/I/2013/3712229275

**ASP.NET XML WEBOVÁ SLUŽBA POSKYTUJÚCA  
INFORMÁCIE O BANKOVÝCH ZMENKÁCH A ICH  
DISKONTOVANÍ**

Diplomová práca

**2013**

**Bc. Lukáš Jancek**

**EKONOMICKÁ UNIVERZITA V BRATISLAVE**

**FAKULTA HOSPODÁRSKEJ INFORMATIKY**

**ASP.NET XML WEBOVÁ SLUŽBA POSKYTUJÚCA  
INFORMÁCIE O BANKOVÝCH ZMENKÁCH A ICH  
DISKONTOVANÍ**

**Diplomová práca**

<b>Študijný program:</b>	Manažérske rozhodovanie a informačné technológie
<b>Študijný odbor:</b>	6258 Kvantitatívne metódy v ekonómii
<b>Školiace pracovisko:</b>	Katedra aplikovanej informatiky
<b>Vedúci záverečnej práce :</b>	Ing. Igor Košťál, PhD.

## **Čestné vyhlásenie**

Čestne vyhlasujem, že na diplomovej práci som pracoval samostatne na základe použitia odbornej literatúry, ktorej úplný prehľad som uviedol v zozname použitej literatúry.

**Dátum: 23.4.2013**

.....  
Lukáš Jancek

## **Pod'akovanie**

Touto cestou by som sa rád poďakoval školiteľovi, pánovi Ing. Igorovi Košťálovi, PhD., za odbornú pomoc a cenné rady pri vytváraní diplomovej práce, a najmä za odborné konzultácie pri tvorbe webovej služby.

## ABSTRAKT

JANCEK, Lukáš: *ASP.NET XML webová služba poskytujúca informácie o bankových zmenkách a ich diskontovaní*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Ing. Igor Košťál, PhD. Bratislava: FHI EU, 2013. 54 s.

Cieľom našej diplomovej práce je vytvoriť ASP.NET XML webovú službu, ktorá bude poskytovať informácie o bankových zmenkách a ich diskontovaní. Vytvoríme funkčnú aplikáciu, ktorá bude vypočítavať obchodný diskont a výsledok bude vracat' v prekonvertovanej forme z XML formátu do HTML formátu. Práca je rozdelená na štyri časti. V prvej sa venujeme histórii xml asp.net technológií a bankovým zmenkám. V druhej kapitole objasňujeme, čo bolo naším cieľom pri dosiahnutí vytvorenia xml webovej služby. Tretia kapitola je venovaná metodike práce a metóde skúmania, a taktiež v nej objasňujeme, aké programovacie jazyky sme využili. V štvrtej kapitole bližšie popisujeme nami vytvorenú webovú službu a spôsob, ako sme postupovali pri jej vytváraní, aby bola správne fungujúca.

**Kľúčové slová:** xml, webová služba, obchodný diskont, banková zmenka, Javascript.

## ABSTRACT

JANCEK, Lukáš: *ASP.NET XML web service providing information about bank acceptance and their discounting*. – University of Economics in Bratislava. Faculty of Business Informatics; Department of Applied Informatics – Supervisor: Ing. Igor Košťál, PhD. Bratislava: FHI EU, 2013. 54 p.

The purpose of our Diploma Thesis is to create the *ASP.NET XML web service* which is providing information about the bank acceptance and their discounting. We will create a functional application that will calculate the trade discounts and the result will be returned back in the converted form from the XML format to the HTML format. Our work is divided into four parts. The first chapter is dedicated to the history of the xml ASP.NET technology and we concentrate on bank acceptance. In the second chapter we explain which was our objective in creating the xml web service. The third chapter deals with the work's methodology and the method of examination and we also analyse there which programming languages we used. In the fourth chapter we describe the web service we created and we also highlight there which manner we used to make it functioning.

**Keywords:** xml, web service, trade discount, bank acceptance, Javascript.

# Obsah

Úvod.....	12
1. História vzniku XML.....	14
1.1. Vznik XML .....	15
1.2. XML súbor .....	15
1.2.1. Pravidlá vytvárania XML štruktúry .....	16
1.2.2. Syntax jazyka XML .....	17
1.3. Kontrola XML dokumentu.....	18
1.4. Transformácia XML.....	19
1.5. ASP.NET technológia .....	20
1.5.1. Dôvody použitia ASP.NET technológie.....	20
1.5.2. Zvýšená spoľahlivosť .....	21
1.5.3. Ľahké zavedenie .....	21
1.5.4. Modely aplikácie.....	22
1.5.5. Produktivita práce .....	22
1.6. Výhody ASP.NET oproti ASP.....	22
1.7. Stavové prostredie nad bezstavovým protokolom .....	23
1.8. Vytvorenie webovej služby.....	23
1.8.1. Vytvorenie webovej služby v Notepade .....	24
1.8.2. Spustenie webovej služby .....	25
1.9. Banková pôžička na zmenku.....	25
2. Ciele práce .....	27
3. Metodika práce a metódy skúmania .....	28
3.1. HTML .....	29

3.1.1.	HTML 5 .....	29
3.2.	Javascript.....	30
3.2.1.	Charakteristika Javascriptu .....	31
3.2.2.	Obmedzenie Javascriptu .....	31
3.3.	AJAX technológia.....	31
3.3.1.	Výhody využívania Ajaxu .....	32
3.4.	Programovací jazyk C# .....	32
3.5.	Kaskádové štýly CSS .....	32
3.5.1.	Vloženie CSS do HTML stránky.....	33
3.5.2.	Selektory v CSS .....	33
3.6.	Využitie HTML, CSS, Javascriptu v práci.....	36
4.	Výsledky práce, vytvorenie XML webová služba.....	38
4.1.	Vloženie zákazníckych údajov.....	39
4.2.	Predčasný odpredaj bankovej zmenky .....	40
4.2.1.	Transformácia dát .....	41
4.2.2.	Rátanie obchodného diskontu .....	43
4.2.3.	Uloženie údajov do XML súboru .....	44
4.2.4.	Vrátenie transformovaných dát do webového klienta .....	45
4.3.	Pridanie novej banky.....	48
4.4.	Prehľadanie XML súboru banka.xml.....	51
4.5.	Celkový sumár odpredaných bankových zmeniek.....	53
4.6.	Usporiadanie bankovej zmenky zostupne .....	54
4.6.1.	Bublínkové triedenie (Bubble sort).....	56
4.7.	Usporiadanie bankovej zmenky vzostupne .....	58
4.8.	Usporiadanie podľa diskontu zostupne .....	61



4.9. Usporiadanie podľa diskontu vzostupne .....	62
Záver .....	64
Zoznam použitej literatúry .....	65

## **Zoznam obrázkov**

Obr. 1: Formulár pre vloženie bankovej zmenky do systému	39
Obr. 2 Rozbaľovacie menu pre vloženie bankovej zmenky	39
Obr. 3 Predčasný odpredaj bankovej zmenky	40
Obr. 4 Vloženie údajov pre prehľadávanie xml súboru vklady	40
Obr. 5 Detské elementy rodičovského elementu zakaznik	43
Obr. 6 Vloženie výsledných údajov do xml súboru zakaznici.xml	45
Obr. 7 Vypočítaný obchodný diskont	46
Obr. 8 Vloženie banky a jej diskontnej sadzby	49
Obr. 9 Užívateľom vloženie banky a jej diskontnej sadzby	49
Obr. 10 XML štruktúra pre vloženie banky	50
Obr. 11 Vrátané transformované dáta z xml súboru po vložení novej banky	51
Obr. 12 Vloženie novej banky do súboru banka.xml	53
Obr. 13 Možnosť výberu usporiadania údajov	53
Obr. 14 Posledné uskutočnené predaje bankových zmeniek	54
Obr. 15 Aktuálny stav posledných uskutočnených transakcií	57
Obr. 16 Usporiadanie bankových zmeniek zostupne	58
Obr. 17 Neusporiadaný zoznam bankových zmeniek	59
Obr. 18 Usporiadanie bankovej zmenky vzostupne	61
Obr. 19 Neusporiadaný zoznam odkúpených bankových zmeniek bankou	61

Obr. 20 Usporiadanie zoznamu podľa diskontu zostupne	62
Obr. 21 Usporiadanie podľa diskontu vzostupne	63

# Úvod

Dôvod výberu témy webovej služby je narastajúca miera používania technológie ASP.NET. V dnešnej dobe sa do popredia dostávajú technológie tvorby webových stránok od spoločnosti Microsoft, ktoré ponúkajú ich vytváranie v jazykoch C#, C++, VB. To bol podnet vybratia tejto témy, keďže sa pohybujeme v oblasti programovania a tvorby webových stránok. Nové technológie a inovácie pri tvorbe dynamických webových stránok sú veľmi dôležitým stavebným materiálom každého programátora. Cieľom práce je vytvoriť funkčnú webovú službu, ktorá umožňuje poskytovať informácie o bankových zmenkách a ich diskontovaní. Pokúsime sa priblížiť hodnotám diskontných sadzieb, teda hodnotám skutočných bánk, ktoré poskytujú diskontné sadzby. Nepôjde však o názvy skutočných bánk. Práca sa skladá z teoretickej a praktickej časti. V teoretickej časti vysvetľujeme a snažíme sa priblížiť technológie, s ktorými pracuje v rámci webovej služby. Druhá časť, praktická, konkrétne objasňuje, ako sme v práci postupovali, a aké techniky a technológie sme použili pri vytváraní našej webovej služby.

V prvej kapitole sa budeme snažiť priblížiť históriu vzniku xml, a tiež to, kto stál za založením a zavedením xml do praxe. Ukážeme si, ako správne vytvárať xml súbor, na čo nesmieme zabudnúť pri jeho vytváraní a aké sú najdôležitejšie pravidlá. Predstavíme si technológiu *ASP.NET*, ktorá je vytvorená spoločnosťou Microsoft. Povieme si o dôvodoch, prečo vytvárať webové stránky v tejto technológii, teda aj o výhodách využitia tejto technológii. Naša diplomová práca sa zaoberá hlavne bankovými zmenkami a ich diskontovaním, preto si povíme, čo je to banková zmenka, kto ju hlavne využíva a prečo ľudia využívajú túto možnosť.

Druhá kapitola je venovaná cieľu diplomovej práce. Bližšie si popíšeme, ako budeme postupovať, aby sme dosiahli náš cieľ. A tým je vytvorenie webovej služby, ktorá bude poskytovať klientovi informácie o bankových zmenkách a ich diskontovaní.

Tretia kapitola popisuje metodiku a metódy skúmania. Popíšeme, ako sme postupovali pri práci, aby sme splnili ciele, ktoré sme si určili v druhej kapitole. Zdôrazníme, čo bolo potrebné využiť a akým spôsobom sme to využili, aký programovací jazyk sme použili a aká technika bola použitá. V práci budeme využívať skriptovací jazyk *Javascript*. Tento jazyk sme v práci využili hlavne pomocou knižnice *JQuery*. Pomocou knižnice *JQuery* sme odchyťovali jednotlivé kliknutia užívateľmi na tlačidlá alebo pre filtrovanie vstupných hodnôt zadávanými užívateľmi. Pomocou knižnice *JQuery* sme

použili aj *AJAX*. *AJAX* je využitý hlavne pri komunikácií medzi klientskym rozhraním a webovou službou. Pomocou *AJAXu* sme posielali dáta z klientskeho okna webovej služby a zase naopak.

V štvrtej kapitole sme rozoberali webovú službu, ktorá je naprogramovaná v jazyku C#. V tejto kapitole je predstavená technika pri tvorbe webovej služby. Postup, ako sme službu vytvárali, aké triedy sme použili pri práci s xml súbormi, bližšie predstavené klientske okno, spôsob ako sú dáta vrátené klientov do prehliadača a spôsob ako sú dáta transformované, aby boli zobrazené užívateľovi vo forme tabuľky. Ukážeme si, ako sa dáta ukladajú už do existujúceho xml súboru. Konkrétne si to predstavíme na pridaní novej banky do xml súboru *banka.xml*. Postupne si budeme prechádzať jednotlivé položky v navigačnom paneli a ukážeme si, ako sú dáta posielané z klientskeho okna do webových metód, ako sa tieto dáta spracúvajú a ukladajú do príslušných xml súborov a naopak, ako sú dáta čítané z xml súboru a odosielané do klientskeho prehliadača v podobe tabuľky.

## 1. História vzniku XML

Prvým motívom, prečo sa začal zavádzať značkovací jazyk, bola potreba štandardizácie. Rozširovanie používania počítačov prinieslo so sebou používanie viac platforiem, operačných systémov a aplikačných riešení. Každý systém a aplikačné riešenie malo vlastný spôsob, ako ukladať dáta, a nie všetky systémy, ktoré spolu komunikovali, boli kompatibilné. Vo firmách a inštitúciách je potrebné, aby dáta boli medzi sebou zdieľateľné. Táto potreba vyvolala možnosť začať štandardizáciu. Avšak veľké giganty si obhajujú svoje záujmy a vytváranie vlastných formátov dokumentov. Cieľ štandardizovať však skôr nahrádzajú cieľmi, aby boli úspešní na marketingovom trhu. Firmy, ktoré vytvárajú desiatky tisíc dokumentov, prakticky nemajú inú možnosť, ako prejsť na iný softvér. Pri problémoch s kompatibilitou sa dokonca stretneme aj pri *MS OFFICE*<sup>1</sup>, preto niektoré firmy nemôžu prejsť na nové softvéry, ale sú nútené dôsledkom nekompatibility využívať zastarané programy. Každá aplikácia si ukladá dáta vo svojom formáte, ktorý je uložený v binárnom kóde, ktorý je pre človeka nečitateľný. Výhodou je hlavne menšia náročnosť na úložný priestor. Snaha o zmenu sa objavila už dávnejšie, kedy predchodcom XML bol jazyk *GML (Generalized Markup Language)*, využívaný spoločnosťou IBM pre ukladanie právnych dát v roku 1960. Podobné snahy začali vznikať aj v iných organizáciách, a tak sa v roku 1980 spojila štandardizačná organizácia *ANSI*<sup>2</sup> spolu so združením *GCA*<sup>3</sup>, aby vytvorili vlastný značkovací jazyk. V roku 1986 vzniká jazyk *SGML (Standard Generalized Markup)*, ktorý je natoľko komplexný, že jeho zložitosť bráni väčšiemu rozšíreniu. Jazyk *SGML* je definovaný v norme ISO 8879<sup>4</sup>. Od jazyka *SGML* sa

---

<sup>1</sup> MS OFFICE je softvér, ktorý bol vytvorený spoločnosťou *Microsoft*. Je využívaný pri práci s dokumentom *MS Word*, s tabuľkami, grafmi *MS Excel*, na tvorbu prezentácií *MS PowerPoint*...

<sup>2</sup> ANSI (*American National Standards Institute*) – Americký národný štandardizačný inštitút.

<sup>3</sup> GCA( *Group Colleges Australia*)

<sup>4</sup> ISO(*International Standards Organisation*)

neskôr odvodil jazyk HTML, dodnes veľmi používaný na tvorbu internetových stránok, ktorý bude ešte veľmi dlho využívaný<sup>5</sup>.

## 1.1. Vznik XML

Dôležitým bodom pre vznik XML jazyka bola potreba oddeliť prezentačnú časť od logickej. Nevýhoda, ktorá vzniká, je obmedzenosť množstva použiteľných značiek, ktoré môžu určitému textu prisúdiť väčší alebo menší význam, ale nemôžu dáta nijakým spôsobom popísať alebo vymedziť. *XML* popisuje len logickú štruktúru, neobsahuje žiadne prezentačné značky a umožňuje jednotlivým blokom textu dať význam nielen hodnotový, ale aj priamo popisuje logický význam dát. Zároveň definuje možnosť stanoviť používateľské značky v dokumente definovanom *DTD*<sup>6</sup>.

## 1.2. XML súbor

XML v preklade znamená *eXtensible Markup Language*, teda rozšíriteľný značkovací jazyk, za ktorým stojí štandardizované konzorcium W3C, ktoré bolo vytvorené z jazyka SGML ako pokračovanie a zároveň zovšeobecnenie značkovacieho jazyka HTML. Poskytuje nám široké spektrum konkrétnych značkovacích jazykov, ktoré môžu byť využité pre rôzne účely, a rovnako poskytuje prácu s rôznymi typovými údajmi. Pomocou XML jazyka aplikácie medzi sebou komunikujú a umožňujú zverejňovanie dokumentov.

Samotný jazyk poskytuje možnosť vytvoriť štruktúru dokumentu z hľadiska obsahu a nekladie dôraz na vzhľad dokumentu, ale na samotné časti. Vzhľad a výzor dokumentu sa upravuje pomocou štýlov. Existujú knižnice, ktoré sú natoľko škálovateľné, že umožňujú transformáciu XML dokumentu.

---

<sup>5</sup> MATĚNA, R.: *XML a značkovací jazyky, historie a vznik*. [online]. Web frequently asked questions. [2008-06-19]. [citované 10. marec 2013]. Dostupné na <<http://www.webfaq.cz/clanek/XML-a-znackovaci-jazyky-historie-a-vznik>>

<sup>6</sup> *DTD (Document Type Definition)* súbor pravidiel, slúžiacich na správne deklarovanie konkrétneho typu dokumentu. Používané hlavne so špecifikáciou *XML*.

Samotný jazyk XML nemá presne definované a určené typové značky, ako je definované v jazyku HTML, ale XML má definované pravidlá, ktoré sa musia dodržať pri správnom vytváraní XML šablóny<sup>7</sup>.

#### HTML štruktúra:

```
<head>
  <title>Toto je jednoduchý titulok html stránky </title>
  <body>
    <h1>Ja som nadpis číslo 1 </h1>
  </body>
</head>
```

#### XML štruktúra

```
<?xml version="1.0" encoding="utf-8"?>
<kniznica>
  <kniha>Základy programovania</kniha>
  <autor>David Morkes</autor>
</kniznica >
```

##### 1.2.1. Pravidlá vytvárania XML štruktúry

Každý vytvorený XML dokument musí dodržiavať zásady vytvárania správneho XML dokumentu. XML dokument je zostavený zo značiek a znakov. Ak sú dodržané zásady a štandardy správneho vytvorenia XML dokumentu, potom štruktúra je považovaná za „well-formed.“ Dokument je považovaný za platný, ak je správne štruktúrovaný a spĺňa DTD požiadavky:

---

<sup>7</sup> Wikipedia. 2013 In: *XML*. [online]. [2013-03-09]. [citované 10. marec 2013]. Dostupné na: <<http://sk.wikipedia.org/w/index.php?title=XML&action=history>>



- Elementy, ktoré sú použité, musia striktne obsahovať začiatočnú a koncovú značku. Nie sú povolené ani prázdne znaky typu `<empty></empty>`. Alternatívou je `<empty/>`. Nepárové znaky `<br/>`, `<img>` nie sú v XML povolené.
- Každý XML dokument musí obsahovať koreňový element, ostatné elementy sa nazývajú detské elementy. Element, ktorý má predchodcu, sa nazýva detský element a predchodcu nazývame rodičovským elementom.
- Aby bol XML súbor správne navrhnutý, je potrebné dodržiavať syntax. Všetky vnorené značky musia byť riadne ukončené.

#### Nesprávny zápis:

```
<?xml version="1.0" encoding="utf-8"?>
<kniha>
  <nazov> Základy programovaniav</kniha>
</nazov>
```

#### Správny zápis:

```
<?xml version="1.0" encoding="utf-8"?>
<kniha>
  <nazov>Základy programovania</nazov>
</kniha>
```

### 1.2.2. Syntax jazyka XML

Zoznam charakteristík, ktoré platia pre XML súbor:

- Je zostavený z elementov a značiek
- Hierarchická usporiadanosť
- Koreňový element je len jeden

- Case sensitive<sup>8</sup>
- Možnosť vnorených elementov
- Medzi dvoma elementami je hodnota elementu
- V elementoch nesmú byť použité značky charakterizujúce komentár <! -- -->
- Elementy obsahujú atribút

### 1.3. Kontrola XML dokumentu

Aby bol XML dokument validný, je potrebné dodržať nasledovné pravidlá:

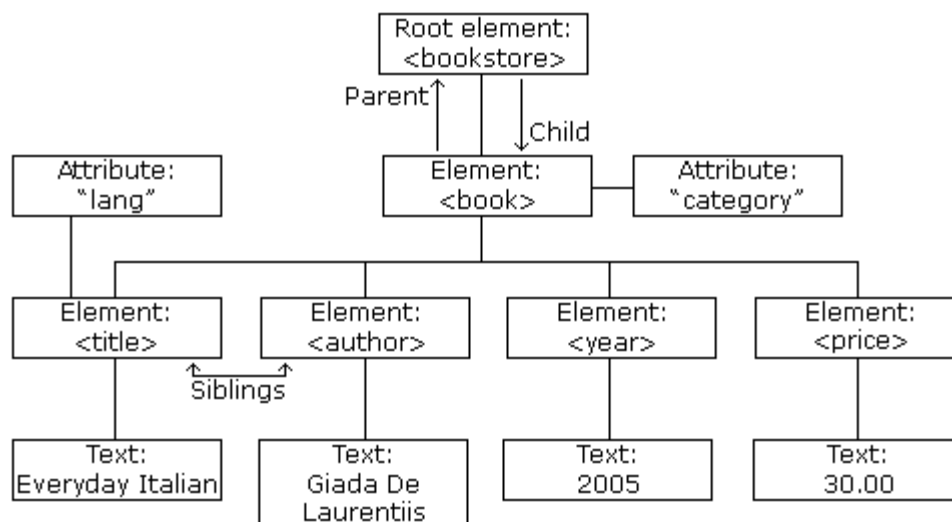
- Koreňový element musí byť striktne jeden
- Názov počiatkovej značky sa musí zhodovať s názvom koncovej
- Zakázané je prekryvať elementy
- Atribúty musia byť umiestnené do úvodzoviek
- Komentáre nemôžu byť použité ako hodnoty elementov

Správne vytvorenie XML dokumentu je možné overiť pomocou:

- Validátora
- Webového rozhrania . IE , Mozilla Firefox
- Parserov

---

<sup>8</sup> Rozlišuje veľkosť písmen. Značka <meno> a <Meno> sú dva rozdielne elementy.



Zobrazenie vzťahov v XML súbore medzi jednotlivými elementami

## 1.4. Transformácia XML

XML umožňuje spôsob, ako transformovať XML štruktúru do tvaru pre klienta zrozumiteľnejšieho. Tento mechanizmus sa nazýva transformácia XSLT (Extensible stylesheet transformation)<sup>9</sup>. Po overení platnosti súboru môžeme pomocou transformácie vytvoriť napríklad marketingovú brožúru v programe Microsoft Office Word 2003. Transformácia sa môže využiť aj na výmenu medzi rôznymi platformami alebo databázami. Pri transformácii XML súboru je základom tzv. template (šablóna). Každý template má definovaný *XPATH*<sup>10</sup>, ktorý prislúcha určitej časti *XML* a obsahuje zoznam pravidiel, ako sa má daná časť transformovať<sup>11</sup>.

<sup>9</sup> XSLT slúži k transformácii dát, ktoré sú v tvare *XML* do ľubovoľného iného tvaru. Najčastejšie táto transformácia je to tvaru *HTML*.

<sup>10</sup> *XPATH* jazyk, ktorý slúži k hľadaniu uzlov v *XML* dokumente.

<sup>11</sup> KOSTELNIK, P.: *Java-XML::how know. Transformácia XML pomocou XSLT*. 2009. [online] [citované 22. marec 2013]. Dostupné na <<http://neuron.tuke.sk/kostelni/tutorialy/xml/xslt-transformation.html>>

## 1.5. ASP.NET technológia

Technológia ASP.NET bola vytvorená spoločnosťou Microsoft a je odvodená od staršej technológie ASP, avšak obe technológie sú od seba odlišné. ASP.NET je postavený na CLR (Common Language Runtime)<sup>12</sup>, ktorý využívajú všetky aplikácie, ktoré sú postavené na .NET Frameworku. Vývojári môžu vyvíjať svoje aplikácie a projekty v akoľvek jazyku využívajúci CLR, napr. Visual Basic.NET, C#, JScript.NET.

„Aplikácie založené na ASP.NET sú rýchlejšie, lebo sú predkompilované do jedného či niekoľko málo DLL<sup>13</sup> súborov, na rozdiel od čisto skriptovacích jazykov, kde sú stránky pri každom prístupne znovu a znovu parsované<sup>14</sup>“.

Technológia ASP.NET umožňuje ľahký prechod od tvorby Windows aplikácií do prostredia web. Aplikácia vytvorená v ASP.NET je kompilátorom preložená do HTML kódu, ktorý sa zobrazí klientovi ako výsledok.

### 1.5.1. Dôvody použitia ASP.NET technológie

Dôvod, prečo je technológia ASP.NET v dnešnej dobe taká žiaduca, je aj zvýšenie výkonnosti a škálovateľnosti. Technológia ASP.NET je svojím výkonom rýchlejšia ako klasické ASP a zároveň ponecháva aktualizácie prostredia ASP. Automaticky rozpoznáva zmenu, dynamicky kompiluje súbory a poskytuje kompilované súbory okamžite po

---

<sup>12</sup> CLR (Common Language Runtime). Rozhranie .NET Framework poskytuje run-time prostredie nazývané CLR. Po spustení kódu, poskytuje službu, ktorá uľahčuje proces vývoja. Nástroje poskytujúce funkcionality modulu CLR umožňujú zapisovanie kódu, ktorý zo spravovaného prostredia po spustení získa výhodu. Kód ktorý je vyvíjaný s kompilátorom jazyka a je zameraný na modul run-time sa nazýva spravovaný kód. Výhoda plynie z integrácie medzi jazykmi, vylepšenie zabezpečenia, podpora pre správu verzií. MSDN. *Spoločný jazykový modul runtime (CLR)*. 2013. [online]. [citované 11. marec 2013]. Dostupné na: <<http://msdn.microsoft.com/cs-cz/library/8bs2ecf4.aspx>>

<sup>13</sup> „Knihnica DLL (DLL) súbor je spustiteľný súbor, ktorý umožňuje programom zdieľať kód a iné zdroje potrebné na vykonanie konkrétnej úlohy“. Microsoft Support. 2013. *Vymedzenie a vysvetlenie. Súbor DLL*. [online]. [citované 12. marec 2013]. Dostupné na: <<http://support.microsoft.com/kb/87934/sk>>

<sup>14</sup> Wikipedia. 2013. In: *ASP.NET*. [online]. [2013-03-15]. [citované 16. marec 2013]. Dostupné na: <<http://sk.wikipedia.org/wiki/ASP.NET>>

požiadavke. Dynamická kompilácia zabezpečuje, že aplikácia je v každom momente aktuálna, a to urýchľuje kompilované prevádzanie. Aplikácie, ktoré prechádzajú z ASP do ASP.NET prostredia, sú schopné obslúžiť tri až päťkrát viac stránok. Ponúkajú aj možnosť ukladania výsledku do medzipamäte. ASP.NET dynamicky zvyšuje výkon aplikácie. V prípade, ak si užívateľ povolí ukladanie stránky do medzipamäte, prostredie ASP.NET vygeneruje stránku len raz a pred odoslaním výsledku je stránka na strane klienta uložená do medzipamäte. V prípade ak iný užívateľ požiada o rovnakú stránku, stránka je načítaná z medzipamäte, bez nutnosti vytvorenia novej stránky.

### *1.5.2. Zvýšená spoľahlivosť*

Technológia ASP.NET poskytuje ochranu proti navráteniu pamäte, zlyhaniu alebo zablokovaniu. Taktiež poskytuje aj automatickú detekciu chýb, ako zablokovanie alebo navrátenie pamäte. Týmto spôsobom sa zaisťuje trvalá dostupnosť aplikácie. V prípade vyskytnutia chyby o návratnosti pamäte, ASP.NET automaticky vyvolá spustenie novej kópie a vyšle požiadavku na nový proces. Starý proces sa po dokončení spracovania požiadaviek čakajúcich na vyradenie odstráni, a navrátená pamäť sa uvoľní.

### *1.5.3. Ľahké zavedenie*

Podobne, ako u iných technológií, rovnako aj technológia ASP.NET je ľahko zavedená. Aplikáciu stačí skopírovať na server. ASP.NET umožňuje aktualizovať skompilované súčasti, ktoré si nutne nevyžadujú reštartovanie webového servera. Po aktualizácii technológia ASP.NET automaticky detekuje zmenu a uvedie do behu nový kód. „V systémoch Microsoft Windows 2000, Windows XP a v systémoch Windows Server 2003 pracuje technológia ASP.NET v službe IBIS súbežne s klasickými aplikáciami prostredia ASP.NET“<sup>15</sup>. Na server je možné prenášať jednu aplikáciu alebo súbory.

---

<sup>15</sup> SHOPcentrik. 2013. *ASP.NET – technológia*. [citované 24. február 2013]. Dostupné na: <<http://www.shopcentrik.sk/slovník/asp-net-technologie-3.aspx>>

#### *1.5.4. Modely aplikácie*

Webové služby XML slúžia k vymieňaniu dát v sieti internet a ku komunikácií aplikácií bez ohľadu na to, na akom operačnom systéme sú spustené alebo v akom programovacom jazyku boli vytvorené. ASP.NET má jednoduchšie volanie webových služieb.

Veľmi rozšírenou možnosťou je podpora mobilných zariadení. Ovládacie prvky smartphonov poskytujú komunikáciu viac než osemdesiatim mobilným webovým zariadeniami, ktoré môžu disponovať ASP.NET technológiou. Mobilné zariadenia automaticky generujú stránku podľa požiadania.

#### *1.5.5. Produktivita práce*

Technológiou ASP.NET sa zvyšuje produktivita práce, čo sa hlavne odzrkadľuje pri vytváraní webových aplikácií. ASP.NET poskytuje veľké množstvo ovládacích prvkov, čo skracuje aj čas a znižuje sa objem kódu ako pri klasickom ASP. Taktiež je aj možnosť komponovať s rôznymi skriptami či už Visual Basic Scripting Edition alebo Microsoft JScript, Visual Basic.NET, Microsoft C# alebo Jscript.NET. Možnosť využívania knižníc pre .NET framework , ktoré ponúkajú viac než 500 tried, v ktorých je zahrnuté množstvo funkcií ako pre jazyk XML alebo prístup k dátam, preposielanie súborov, regulárne výrazy, generovanie obrázkov, poslanie emailu protokolom SMTP a množstvo ďalších.

### **1.6. Výhody ASP.NET oproti ASP**

- Kód je kompilovaný, aplikácia je rýchlejšia a chyby sú odchytené už pri vývoji
- Možnosť využitia šablón, redukcia duplicitného kódu
- Široká škála ovládacích prvkov (button, label, canvas )
- Knižnice, ktoré zrýchľujú vývoj aplikácií
- Viac programovacích jazykov

- Možnosť „nacacheovania“<sup>16</sup> stránky, čo umožňuje rýchlejšie načítanie stránky
- Možnosť behu na rôznych serveroch (IIS, Apache)

## 1.7. Stavové prostredie nad bezstavovým protokolom

„Aj keď webový protokol *HTTP*<sup>17</sup> je sám o sebe bezstavový (t.j. jednotlivé požiadavky od užívateľa medzi sebou nie sú previazané), vyžaduje zachovanie kontextu medzi jednotlivými požiadavkami. ASP.NET tento problém rieši kombináciou HTML a JavaScriptu pomocou dvoch techník *ViewState* a *Session State*.“<sup>18</sup>

## 1.8. Vytvorenie webovej služby

K vytvoreniu webovej služby stačí obyčajný textový editor, napr. Notepad a nainštalovaný .NET Framework. Najlepšie prostredie nám poskytuje Visual Studio. Ďalšou dôležitou časťou je mať server, kde webovú službu uložíme, a server, ktorý bude podporovať *ASP.NET*, napr. *IIS*.

---

<sup>16</sup> *Cache* stránky je spôsob ako urýchliť načítanie stránky do prehliadača. Táto možnosť sa využíva hlavne u webových stránkach, ktoré načítavajú veľké množstvo údajov hlavne pri čítaní údajov z databázy. Princíp spočíva v tom, že stránka je uložená na server v *html* formáte a pri načítaní stránky je načítaný len obsah *html*. Aby sa nemuseli vykonávať veľké množstvo dotazov na databázu sa týmto spôsobom odľahčuje stránka a je skôr zobrazená klientovi.

<sup>17</sup> *HTTP* (HyperText Transfer Protocol) je najpoužívanejší protokol na internete. Pomocou *http* protokola sa dáta odosielať z webového servera do klientskeho prehliadača. Činnosť protokolu môžeme definovať nasledovne: po naviazaní spojenia so serverom napr. ([www.lukasjancek.eu](http://www.lukasjancek.eu)) klient (Netscape, Lynx) odošle požiadavku napr. *GET /stranka.html http/1.1*. Okrem tohto parametre sú potrebné aj ďalšie parametre tie však neuvádzame. Prvý parameter *GET* udáva aká metóda bude použitá. Môžeme použiť napríklad spomínanú *GET* metódu alebo *POST* / *HEAD* metódu. Druhá časť *stranka.html* znamená cesta k súboru na servery, čo znamená ktorý súbor sa bude čítať.

<sup>18</sup> Wikipedia. 2013. In: *ASP.NET*. [online]. [2013-03-15]. [citované 17. marec 2013]. Dostupné na: <<http://sk.wikipedia.org/wiki/ASP.NET>>

### 1.8.1. Vytvorenie webovej služby v Notepade

Na vytvorenie správne fungujúcej webovej služby je potrebné, aby súbory, ktoré tvoria webovú službu, mali správnu príponu .asmx. Prázdny textový súbor sa uloží pod názvom nazov.asmx . Do súboru sa zapíše logika celej webovej služby. Webová služba bude sčítavať dve celé čísla . Nasledovný kód bude vytvorený v jazyku C#.

```
public Class Sucet{  
  
    public int Sucet(int a,intb){  
  
        return a+b;  
  
    }  
  
}
```

Pre sprístupnenie metódy *Sucet na web* je potrebné pridať atribút *WebMethod*. Atribúty sú deklarované kódom, ktorý pridáva správanie. Metódy a vlastnosti označované atribútom *WebMethod* sú prístupné ako webová služba. Môžu prijímať a odosielať XML správy, pričom o serializáciu <sup>19</sup>a deserializáciu nie je potrebné sa starať.

Aby webová služba fungovala, je potrebné zahrnúť direktívu, ktorá informuje o tom, že vytvárame webovú službu, aký programovací jazyk je používaný pre kódovanie aplikačnej logiky a ktorá trieda má označenie metódy atribútom *WebMethod*. Úplný kód vyzerá nasledovne :

```
<%@ WebService Language="C#" Class=" ScitanieDvochCisiel "%>
```

```
using System;
```

```
using System.Web.Services;
```

---

<sup>19</sup> Pri behu programu sú dáta ukladané do pamäti počítača. Pre uloženie dát do súboru alebo pre posielanie cez sieť, je potrebné použiť vhodnú formu štrukturalizácie. Proces akým spôsobom sú štruktúrované dáta zapísané z pamäte počítača do streamu sa nazýva serializácia. Naopak zas čítanie zo streamu do pamäte sa nazýva deserializácia. ANDRÁSSY, J.: *Serializácia štruktúrovaných dát*. [online]. [2007-11-14]. [citované 26. marec 2013]. Dostupné na: <<http://www.vsl.sk/vyuka/java07/XML.pdf>>



```

public class ScitanieDvochCisiel {

    [WebMethod]

    public int Scitaj (int a, int b) {

        return a + b;

    }

```

### 1.8.2. Spustenie webovej služby

Aby webová služba fungovala, musí byť umiestnená na webovom serveri, ktorý podporuje ASP.NET. Pre testovanie funkčnosti stačí webový prehliadač. Kompilácia nie je potrebná, pretože kompilácia prebehne pri prvej požiadavke na webovú službu.

## 1.9. Banková pôžička na zmenku

Podľa expertov je „pôžička na zmenku v posledných rokoch najvyhľadávanejším a najrýchlejším spôsobom, ako získať peniaze v hotovosti a okamžite bez dlhšieho čakania na vybavenie pôžičky.“<sup>20</sup> Klienti využívajú túto možnosť hlavne vtedy, keď už majú v banke úver a ďalšia pôžička im nemôže byť vystavená z tohto dôvodu. Pri bankových zmenkách však nie sú bankové registre kontrolované. Ako ďalej dodávajú finanční experti, „ak má klient záznam v registri dlžníkov, je takmer nemožné získať pôžičku od banky. Vtedy prichádzajú na rad nebankové spoločnosti, ktoré čakajú práve na takýchto klientov. Vďaka menším požiadavkám si totiž môžu dovoliť väčšie úroky a poplatky, a teda majú aj väčšie zisky. Jednou z možností voľby pôžičiek u týchto spoločností je pôžička na zmenku ihneď a bez registra“.<sup>21</sup> „Potom stačí len zariadiť bankový účet, z ktorého bude

---

<sup>20</sup> Banková pôžička na zmenku. 2009. [online]. [citované 2. apríl 2013]. Dostupné na: <<http://bankova-pozicka.webnode.sk/bankove-pozicka-na-zmenku/>>

<sup>21</sup> Pôžičky na zmenku ihneď a bez registra. [online]. [citované 2. apríl 2013]. Dostupné na: <<http://uveryapozicky.sk/pozicky-na-zmenku-ihned-a-bez-registra>>

suma čerpaná a prebehne zistenie exekučného stavu na Vašu osobu, či napríklad sa na Váš plat nevzťahujú exekúcie a pod. Pre bližšie informácie si stačí prejsť internetové stránky bánk a nebankových poskytovateľov pôžičiek na zmenku, porovnať si požiadavky a nezáväzne sa informovať treba pomocou online formulára.“<sup>22</sup>

---

<sup>22</sup> *Banková pôžička na zmenku*. 2009. [online]. [citované 2. apríl 2013]. Dostupné na <<http://bankova-pozicka.webnode.sk/bankove-pozicka-na-zmenku/>>

## 2. Ciele práce

Predmetom riešenia našej diplomovej práce je vytvorenie funkčnej webovej služby, ktorá bude poskytovať informácie o bankových zmenkách a ich diskontovaní. Cieľom je vypočítať obchodný diskont na základe fiktívnych bánk a ich diskontných sadzieb, ktoré sa približne zhodujú s inými bankami a ich diskontnými sadzbami. Webová služba vyráta, aký obchodný diskont bude zrazený z bankovej zmenky pri predčasnom odpredaní. Ďalším cieľom pri vytváraní webovej služby bolo, aby užívateľ dostal do klientskeho okna transformované dáta, pretože webová služba dáta vráti v podobe xml. Tieto dáta sú náročné pre bežného užívateľa, preto bolo potrebné dáta transformovať.

### 3. Metodika práce a metódy skúmania

V metodike práce dôkladne popíšeme spôsob, ako sme vytvárali webovú službu, aké techniky a technológie sme použili pre vytvorenie správne fungujúcej webovej služby. Business model nami vytvorenej webovej služby pozostáva z troch častí. Jednotlivé časti sú medzi sebou prepojené a vzájomne si posielajú dáta.

Prvou časťou je klientske okno. V klientskom okne sú užívateľovi zobrazené možnosti, ako výška bankovej zmenky, vloženie novej banky, predčasný odpredaj bankovej zmenky a celkový sumár. Klientske okno webového prehliadača je čisto vytvorené len v html, s použitím kaskádových štýlov a pomocou Javascriptu. Ďalšou výhodou je aj možnosť si pozrieť bližšie informácie o webovej službe. V päte stránky po kliknutí na text *Viac*, sú mu zobrazené podrobnejšie informácie o webovom klientovi, bližšie informácie o použitých technikách.

Druhú časť tvorí samotná webová služba. Webová služba tvorí jadro celej aplikácií. Webová služba je vytvorená v jazyku C#. Služba obsahuje jednu triedu `WebService_XML_file`. Obsahuje súkromné premenné, v ktorých definujeme relatívnu cestu ku xml súborom. V konštruktoe sú načítané všetky xml súbory, aby sme nemuseli vždy načítavať opakovane do pamäti, sú načítane vždy raz pri každom spustení webovej služby. Každá metóda webovej služby musí mať definovanú hlavičku, ktorá dokazuje, že je to webová metóda. Vždy pri posielaní údajov do klientskeho okna z webovej metódy sú dáta transformované. Aby webová služba bola dynamická, použili sme Ajax, ktorý nenačítava celý obsah stránky, ale len určitú časť, a tým sa aj znižuje zaťaženie.

Webové metódy prijímajú dáta, ktoré pošle webový prehliadač pomocou Ajaxu do metód, ktoré sú definované vo webovej službe, ktoré preberajú dáta ako parameter a následne tieto dáta uloží do príslušných xml súborov. Pomocou webových metód sú dáta čítané, zapisované do xml súborov, vykonáva sa usporiadanie dát, vypočítavanie obchodného diskontu z vlastnených bankových zmeniek. Po prijatí dát webovými metódami od klientskeho prehliadača sú vykonané operácie s týmito údajmi, následne sa zapíšu údaje do príslušného xml súboru, potom sú dáta načítané z xml súboru webovou službou, ďalej sú odoslané do klientskeho okna, ktoré ich zobrazí. V aplikácii sme využili aj zapisovanie do už existujúceho xml súboru. Postup zapisovania prebieha tak, že nové dáta sa zapíšu vždy na koniec súboru. Existujúci xml súbor je načítaný a na koniec sa pridávajú nové dáta.

Poslednou časťou je samotný úložný priestor na dáta. Pri jeho tvorbe sme použili tri xml súbory. Xml súbory banka.xml, vklady.xml, zakaznici.xml.

### 3.1. HTML

Jazyk *HTML* (*HyperText Markup Language*) je značkovací jazyk, ktorý je predovšetkým využívaný k tvorbe jednoduchých webových stránok. Jazyk *HTML* je využívaný hlavne na prezentovanie informácií do blokov, odsekov a tabuliek. Jazyk *HTML* sa ujal hlavne pre jeho jednoduchosť. Definícia jazyka je vymedzená obmedzenou množinou elementov, ktoré určujú štruktúru dokumentov. Značkovací jazyk *HTML* bol prvotne určený ako podmnožina jazyka *SGML*, až neskôr sa stal samotným štandardom. Pod jazyk *HTML* a jeho špecifikáciu sa radí konzorcium W3C (World Wide Web Consortium)<sup>23</sup>. Najnovšia verzia *HTML* je *HTML 5*, v ktorej pribudli nové elementy ako v jazyku *HTML 4*.

#### 3.1.1. HTML 5

*HTML 5.0* je momentálne najaktuálnejšou verziou zo všetkých *HTML*. *HTML 5* sa dostáva stále do popredia hlavne v spojení s *CSS 3*, kde je možné pomocou moderných prvkov nahradiť *flash*<sup>24</sup> animácie. *HTML 5* sa od staršej verzie *HTML 4* líši napr:

- Rýchlejším zápisom značiek
- Možnosť fungovania aplikácií v offline režime
- Pribudli nové značky, napr: prehrávanie videa priamo v prehliadači.
- Možnosť vytvorenia aplikácie , ktorá bude fungovať aj bez pripojenia na internet.

---

<sup>23</sup> W3C je konzorcium, ktoré okrem zavádzania nových webových technológií, núti webových vývojárov aby dodržiavali štandardy, ktoré sú určené a tým aj smerovali ku priaznivému vývoju webu. Pri tvorbe web stránok by sa mal dodržiavať štandard, ktorý určuje práve toto konzorcium. To či je stránka validná alebo nie je možné zistiť pomocou validátora či už na (x)html alebo css.

<sup>24</sup> *Flash* je program od firmy *MACROMEDIA* , ktorý je určený predovšetkým na tvorbu animácií. Je založený na vektorovej grafike.

Príchodom HTML 5 prichádza aj nová špecifikácia zápisu pre typ dokumentu. Zápis sa veľmi skrátil a odstránila sa aj *DTD* špecifikácia. Začínajúci dokument v HTML 5 stačí napísať nasledovne :

```
<!DOCTYPE html>
```

V programoch, kde stránky boli vytvorené v inej verzii ako HTML 5, bolo potrebné definovať úplnú hlavičku HTML dokumentu. Napríklad tak, ako je to pri HTML 4:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Ďalšou vecou, na ktorú programátori mysleli, je sémantika, čo zvýši hlavne prehľadnosť kódu. Väčšina programátorov si stránku pri jej tvorbe pri tvorbe rozdelí do niekoľkých logických blokov, ako napr.: hlavička, telo, päta, menu. V HTML 4 bolo potrebné pomocou značky *div* a identifikátora nastavenia týchto blokov. Napr.:

```
<div id="hlavicka"></div><div id="pata"></div>
```

Týmto spôsobom sa identifikovalo, že prvý div je hlavička webovej stránky a druhý div je päta webovej stránky. V HTML 5 mysleli na toto oddelenie a pridali nové značky, a to:

```
<header></header><footer></footer>
```

Okrem týchto dvoch značiek pribudli aj ďalšie, ako *video*, ktoré umožňuje vložiť video na webovú stránku alebo *audio*, teda vloženie zvukovej stopy.

## 3.2. Javascript

*Javascript* patrí medzi programovacie jazyky, ktoré sú najviac využívané v internetových webových stránkach a internetových aplikáciách. Môže byť zapísaný priamo v html kóde, v ktorom je potrebné uviesť hlavičku zapisovania syntaxe: `<script>` //kód `</script>`, alebo v externom súbore. Javascript sa radí medzi klientske skripty,

pretože program je odosielaný do prehliadača klienta, a tam je aj vykonávaný. Naproti klientskym skriptom sú serverové skripty, ktoré sú výlučne vykonávané na strane servera.

### *3.2.1. Charakteristika Javascriptu*

- Interpretovaný – nie je nutnosť kompilácie
- Objektový – možnosť vytvárania objektov alebo využívanie objektov prehliadača
- Závislý na prehliadači – funguje vo väčšine prehliadačov
- Case sensitive – nutnosť dodržiavania veľkých a malých písmen
- Syntax je podobná jazykom Java, C

### *3.2.2. Obmedzenie Javascriptu*

- Javascript funguje výlučne v prehliadači
- Možnosť povolenia a zakázania Javascriptu
- Rôzne verzie prehliadačov
- Nemožnosť pristupovať k súborom
- Neumožňuje uloženie dát, okrem cookies <sup>25</sup>

## **3.3. AJAX technológia**

*AJAX* znamená *Asynchronous JavaScript + XML* a nepatrí medzi samostatné programovacie jazyky ani technológiu. Ide o kombináciu Javascriptu XML a skriptovacieho jazyka na strane servera. Využívajú sa hlavne pri webových stránkach, aby umožňovali dynamicky meniť obsah stránky bez kompletného načítania stránky zo servera.

---

<sup>25</sup> JANOVSKEÝ, D.: *Čo je to Javascript*. [online]. Jak psat web. [2012-12-06]. [citované 3. apríl 2013]. Dostupné na: <<http://www.jakpsatweb.cz/javascript/javascript-uvod.html>>

### 3.3.1. Výhody využívania Ajaxu

Hlavnou výhodou použitia Ajaxu je podstatné urýchlenie behu stránky a internetových aplikácií, pretože nie je potrebné vždy načítať celý obsah stránky opakovane, čím sa funkcionálnosť a rýchlosť stránky zvyšuje. Medzi ďalšie výhody patrí úspora prenosovej kapacity, pretože kód stránky nie je zakaždým posielaný. Pri dynamických stránkach je posielaná len časť stránky, ktorá sa má meniť.<sup>26</sup>

## 3.4. Programovací jazyk C#

Programovací jazyk C# sa radí medzi objektovo orientované programovacie jazyky. Za vývojom stojí americká spoločnosť Microsoft. Hlavným konštruktérom bol *Anders Hejlsberg*. Jazyk C# bol vyvinutý z jazykov C++ a Javy. V dnešnej dobe sa radí medzi obľúbené programovacie jazyky, hlavne vďaka svojej jednoduchosti a ľahkosti naučenia. Pri vytváraní jazyka C# vývojári kladli dôraz na to, aby jeho implementácia poskytovala podporu softvérového inžinierstva, silnú typovú kontrolu, ohraničenosť polí, správu pamäte ako automatickú a detekciu premenných, ktoré nie sú inicializované. Programátor so znalosťami z jazykov C a C++ by nemal mať veľké problémy pri prechode na C#, a rovnako aj možnosť prenositeľnosti zdrojového kódu. Pri vývoji bolo cieľom aj ekonomické využitie pamäte. Nie je zameraný do takých detailov ako sú jazyky C alebo assembler, čo sa týka výkonnosti a rovnako aj veľkosti binárneho kódu.

## 3.5. Kaskádové štýly CSS

CSS štýly nazývané kaskádovými štýlmi sa používajú k vytváraniu štýlov na stránke HTML. Pomocou CSS môžeme použiť napríklad farebné pozadie na stránke, či už v jednej farbe, napr. čiernej, bielej alebo môžeme použiť obrázkov, alebo prechody.

---

<sup>26</sup> LACKO, Ľ.: *Výhody a nevýhody technológie AJAX*. [online]. Infoware. [2009-12-07]. [citované 10. apríl 2013]. Dostupné na <<http://www.itnews.sk/tituly/infoware/2009-12-07/c130666--vyhody-a-nevyhody-technologie-ajax>>



Pomocou jedného súboru je možné ovplyvniť vzhľad celej webovej stránky. CSS (v preklade *Cascading Style Sheets*) sa prvý krát implementovalo spoločnosťou Microsoft do Internet Explorera 3.0 v roku 1996. Okrem spomínaných vlastností stránky, ako pozadie, je možné ovplyvniť ďalšie vlastnosti, ako napríklad písmo, rámčeky, tabuľky, odrážky, okraje a mnoho ďalších. CSS je hlavne aplikovaný pomocou selektorov a tried. Pomocou selektorov a tried môžeme vytvárať atribúty a priradovať ich ku ľubovoľným elementom bez toho, aby sme museli opakovať dané príkazy.

### 3.5.1. Vloženie CSS do HTML stránky

Vloženie CSS do html stránky je možné dvomi spôsobmi. Prvý spôsob je priamo do html kódu do hlavičky, medzi značky <head></head> vpísať jednoduché značky <style></style>, ktoré definujú, že sa jedná o css. Druhým spôsobom, ako je možné na stránku pridať css, je vloženie externého súboru. Externý súbor musí mať svoj názov a príponu .css *style.css*. Do html stránky do hlavičky vpíšeme nasledovný kód, ktorý vloží css do html stránky:

```
<head>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

### 3.5.2. Selektory v CSS

Selektory v CSS majú veľmi dôležitú úlohu, pretože pridávajú jednotlivým prvkom na stránke vlastnosti. Prvým spôsobom, ako je pridať vlastnosti prvku, je pomocou názvu prvku, napr.:

```
h1{ text-decoration: underline;font-size: 12px; color: red;}
```

Ak na stránke máme viac nadpisov typu h1, h2, h3 a chceme každému nadpisu dať rovnakú vlastnosť, môžeme to urobiť nasledovným spôsobom:

*h1,h2,h3{ text-decoration: underline;font-size: 12px; color: red;}*

Týmto spôsobom sa zabezpečí, že nadpisy budú mať všetky rovnaké vlastnosti. Je ošetrované, že nemusíme rovnaké vlastnosti vždy vypisovať pre každý nadpis zvlášť.

Potomok, alebo kontextový selektor je selektor, ktorý pridá vlastnosti prvku len v prípade, ak sa nachádza v určitom prvku. Príklad kontextového elementu môže byť nasledovný:

```
<div><p>diplomova praca</p></div>
```

Párová značka *p*, sa nachádza vnútri párovej značky *div*, pre jej nastavenie vlastnosti, môžeme selektor napísať nasledovne:

```
div p {color: red;}
```

Kontextové selektory môžu byť rôzne. Následnosť prvkov záleží od samotného programátora, ako si navrhne štruktúru. Musí byť len dodržaná následnosť, napr:

```
<ul>
<li><p>Prvy</p></li>
<li><p>Druhy</p></li>
</ul>
```

Aby sme nastavili farbu textom *Prvy* a *Druhy* musíme dodržať následnosť presne v tom poradí, v akom sme napísali fragment kódu:

```
ul li p {color> red;}
```

Ďalším spôsobom, ako je možné identifikovať, akej značke chceme nastaviť štýl, je možné pridať značke triedu. V *html* sa do značky pridá atribút *class*, napr.:

```
<div class="content"></div>
```

Ak chceme nastaviť štylovanie *divu*, ktorému sme priradili triedu *content*, musíme v *css* definovať štýl triede *content*, a to priradením bodky pred názov:

```
.content{width: 30px;height: 30px; border: 1px solid black;}
```

Divu *content* sme nastavili výšku a šírku 30 pixelov a orámovanie bude čierne.

Podobne ako triedy je možné použiť aj identifikátor *id*. V *html* súbore je potrebné ku značke pridať len názov *id*, napr: `<div id="content"></div>` . V CSS súbore už len stačí pred názov *content* pridať `#` .

*#content{width: 30px; height: 30px; border: 1px solid black;}*

V *css* je možnosť aj kombinovať triedy s identifikátormi, pridávať viac tried, napr.:

*HTML* kód

```
<div id="content" class="pozadie"></div>
```

```
<div id="content" class="pozadie oramovanie"></div>
```

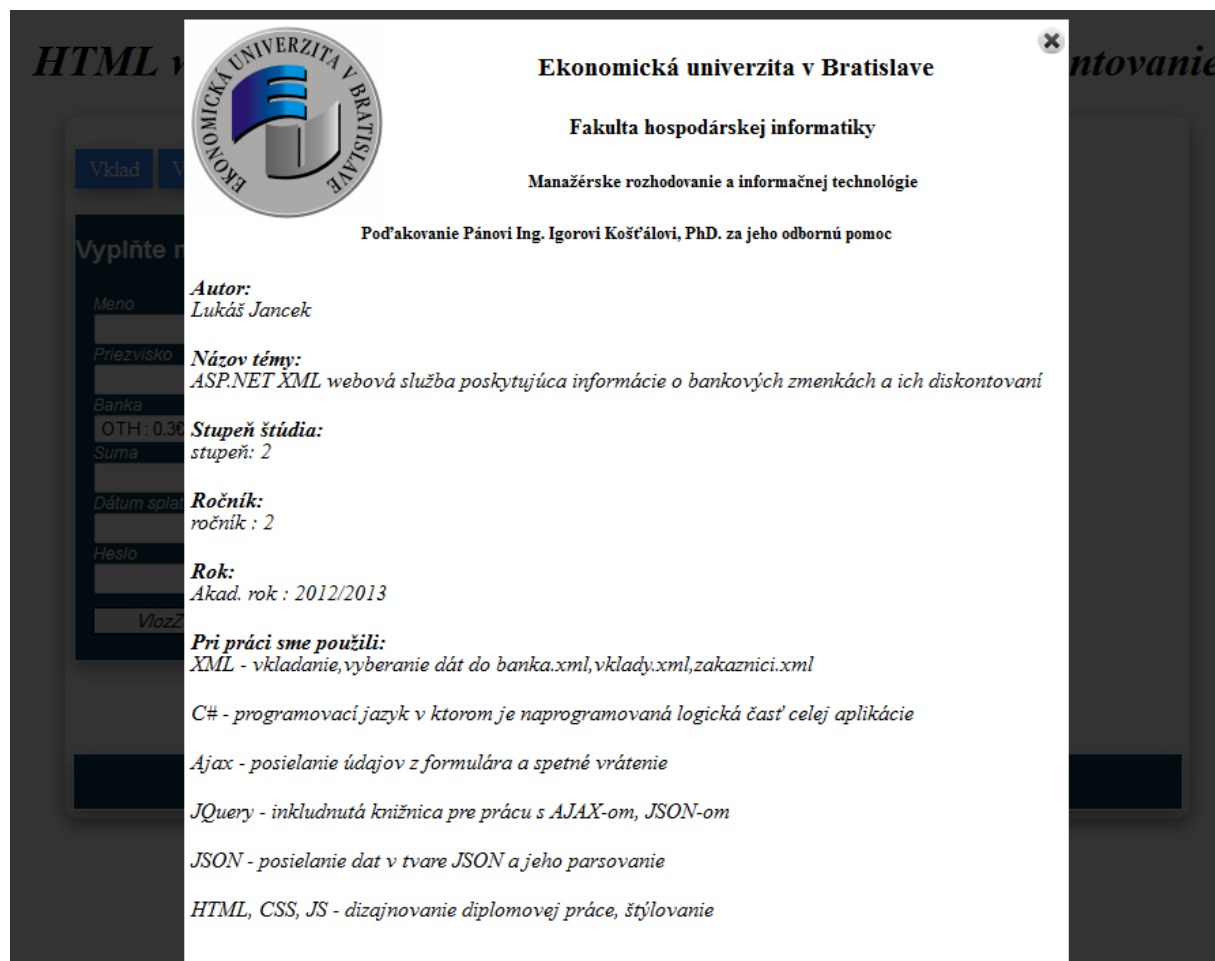
Css kód môžeme zapísať viacerými spôsobmi, oba však vykonajú rovnaké operácie:

#content { height: 30px; width:30px; }	div#content{ height: 30px; width:30px; }
.pozadie { background-color: red; }	div.pozadie{ background-color: red; }
.oramovanie{ border: 1px solid black; }	div.oramovanie{ border: 1px solid black; }

Obe značky *div*, budú mať rovnakú výšku, rovnakú šírku a aj pozadie, ale druhý *div* bude mať rámovanie čierne, a prvý nie.

### 3.6. Využitie HTML, CSS, Javascriptu v práci

Jazyk HTML bol nami vybraný pre vytvorenie klientskeho okna. Pri spustení klientskeho okna klientom je možné si prečítať základne informácie o nás. Rovnako aj informácie o vysokej škole, fakulte a kto bol školiteľom práce. Na tomto obrázku je možné vidieť, ako sme skĺbili všetky tri jazyky dokopy. V päte stránky je na konci slovo *viac*. Po kliknutí na slovo *viac* sa nám zobrazí nasledovný obrázok:



**Ekonomická univerzita v Bratislave**  
**Fakulta hospodárskej informatiky**  
**Manažérske rozhodovanie a informačnej technológie**

PodĎakovanie Pánovi Ing. Igorovi Košťálovi, PhD. za jeho odbornú pomoc

**Autor:**  
Lukáš Jancek

**Názov témy:**  
ASP.NET XML webová služba poskytujúca informácie o bankových zmenkách a ich diskontovaní

**Stupeň štúdia:**  
stupeň: 2

**Ročník:**  
ročník : 2

**Rok:**  
Akad. rok : 2012/2013

**Pri práci sme použili:**  
XML - vkladanie, vyberanie dát do banka.xml, vklady.xml, zakaznici.xml  
C# - programovací jazyk v ktorom je naprogramovaná logická časť celej aplikácie  
Ajax - posielanie údajov z formulára a spetné vrátenie  
JQuery - inkludnutá knižnica pre prácu s AJAX-om, JSON-om  
JSON - posielanie dát v tvare JSON a jeho parsovanie  
HTML, CSS, JS - dizajnovanie diplomovej práce, štylovanie

Vyskakovacie okno s bližšími údajmi k diplomovej práci

- *HTML* sme použili na výpis údajov do blokov a vloženie loga Ekonomickej univerzity.
- *CSS* bolo použité na dizajnovanie okna.
- *Javascript* bol použitý na nastavenie viditeľnosti okna. Najskôr je okno skryté pomocou css príkazu `visibility : hidden`. Po kliknutí myšou sa nastaví tak, aby bola vykonaná operácia `visibility: visible`, a okno sa zobrazí v prehliadači. Pre vypnutie vyskakovacieho okna stačí kliknúť na krížik, ktorý sa nachádza v pravom hornom rohu a vyskakovacie okno je znova nastavené na skryté.

Nie len na zobrazenie vyskakovacieho okna sme použili skĺbenie *html*, *css* a *Javascriptu*, ale aj na výpis údajov, ktoré sú vrátené z webových metód. Vždy pri vrátení údajov, ktoré sú poslané webovou metódou, sa tieto údaje formátujú, aby boli čo najľahšie čitateľné koncovým užívateľom. Údaje sú vpisované do tabuliek, čo je asi najľahší spôsob, ako prehľadne zobraziť dáta. Stĺpce aj riadky tabuľky majú nastavené pozadie, aby sa užívateľ cítil na stránke príjemne.

## 4. Výsledky práce, vytvorenie XML webová služba

Vývojové prostredie pre vytvorenie xml webovej služby, ktorá poskytuje informácie o bankových zmenkách a ich diskontovaní, je prostredie Visual Studio<sup>27</sup> od spoločnosti Microsoft a programovací jazyk, v ktorom je aplikácia vytvorená, je jazyk C#. Pre spustenie webového klienta je potrebné mať nainštalovaný internetový prehliadač, napr. Mozilla Firefox, Opera, Chrome, najlepšie s najaktuálnejšou verziou a mať povolené Javascripty. Webová služba umožňuje vkladať už do vytvoreného xml súboru údaje, vyberanie údajov podľa parametra a usporiadať vybrané údaje. Služba pracuje s nasledujúcimi xml súbormi

- Banka.xml
- Vklady.xml
- Zakaznici.xml

*Banka.xml* – xml súbor, do ktoré sú ukladané údaje o názve bánk a ich diskontných sadzbách. Názvy bánk nekorešponujú so žiadnou reálnou bankou, a rovnako ani diskontné sadzby.

*Vklady.xml* – xml súbor do ktorého sú ukladané údaje o zákazníkoch. Meno, priezvisko zákazníka, vybratá banka a jej diskontná sadzba, výška zmenky, dátum splatnosti a heslo.

*Zakaznici.xml* – xml súbor, do ktorého sa ukladajú údaje o výške bankovej zmenky, ktorú zákazník vlastní a údaje o jej následnom skoršom odpredaní, vypočítaní výšky obchodného diskontu, ktorý bude zrazený, a suma, ktorá bude vyplatená zákazníkovi.

Webová služba je zložená zo štyroch častí:

- Vloženie zákazníckych údajov
- Predčasný odpredaj bankovej zmenky
- Pridanie banky do systému

---

<sup>27</sup> Visual studio je prostredie, ktoré bolo vytvorené americkou spoločnosťou Microsoft za účelom vytvárania softvérových aplikácií na najnovších platformách pomocou moderných nástrojov pri riadení životných cyklov. Hlavnej v najnovšej verzii Visual studio 2012.

- Súhrnné informácie o všetkých bankových zmenkách

#### 4.1. Vloženie zákazníckych údajov

Vypĺňte nasledovný formulár

Meno

Priezvisko

Banka  
 OTH : 0.3€ ▼

Suma

Dátum splatnosti

Heslo

VlozZakaznika

Obr. 1: Formulár pre vloženie bankovej zmenky do systému

Po otvorení webového klienta, je užívateľovi zobrazený formulár, kde je povinné vyplniť všetky vstupné polia: (meno, priezvisko,...). Z ponúknutého rozbaľovacieho okna si klient vyberie banku s ponúkanou diskontnou sadzbou, od ktorej vlastní bankovú zmenku, následne vloží výšku vlastniacej bankovej zmenky v eurách,

OTH : 0.3€ ▼

OTH : 0.3€

UHBank : 0.8€

UKH Group : 0.74€

SSH : 0.09€

SVST : 0.07€

KONTROLA : 0.09€

posledna : 0.57€

Streda : 0.09€

AHA : 0.04€

DEXIA : 0.11€

Obr. 2 Rozbaľovacie menu pre vloženie bankovej zmenky

dátum splatnosti uvedie v tvare deň.mesiac.rok a ľubovoľný reťazec (heslo), ktorý bude identifikovať každého jedného zákazníka. Po kliknutí na tlačidlo „VlozZakaznika” sú dáta uložené do XML súboru s názvom vklady.xml .

#### 4.2. Predčasný odpredaj bankovej zmenky

Vyplňte nasledovný formulár

Heslo

Dátum výberu

Vyhľadaj zákazníka

Obr. 3 Predčasný odpredaj bankovej zmenky

Užívateľ do textového poľa vloží *Heslo*: vpíše heslo, ktoré zadal pri vkladaní údajov do systému a vyberie si dátum, kedy chce svoju vlastniacu bankovú zmenku odpredať. Dátum zadá v nasledovnom tvare : deň.mesiac.rok .

Vyplňte nasledovný formulár

Heslo

LA

Dátum výberu

29.5.2013

Vyhľadaj zákazníka

Obr. 4 Vloženie údajov pre prehľadávanie xml súboru vklady



### 4.2.1. Transformácia dát

Na transformáciu dát je využitá knižnica jquery, ajax. Po kliknutí na tlačidlo *Vyhľadaj zákazníka* sú dáta serializované do premennej *data* z formulára s id *main\_form* pomocou jquery knižnice.

```
var data = $("#main_form").serialize();
```

*Jquery knižnica* ponúka funkciu *serialize()*, ktorá dáta z formulára vloží do reťazca. Reťazec je následne poslaný pomocou AJAX-u do patričnej metódy webovej služby. Webová metóda, do ktorej sa majú odoslať dáta, je v parametri *url*.

```
$.ajax({
    type: "POST",
    contentType: "application/json; charset=utf-8",
    url: "WebService_XML_file.aspx/VlozDoXMLSuboru",
    data: '{udaje: "' + data + '" }',
    dataType: "json",
});
```

- *type*: určuje, že dáta budú poslané pomocou metódy POST
- *contentType*: nastavenie kódovania posielaných dát
- *url*: nastavenie funkcie, do ktorej budú dáta poslané *ajaxom*
- *dataType*: dáta sú posielané v *json reťazci*
- *data*: do premennej *udaje* je uložený obsah, čo sa má poslať do webovej metódy

V súbore s názvom *WebService\_XML\_file.aspx.cs* je metóda *VlozDoXMLSubore*, ktorá ma vstupný parameter *string udaje*, do ktorého sa uloží reťazec poslaný *ajaxom*:

```
public string VlozDoXMLSuboru(string udaje){}
```

Pred každou funkciou je potrebné uviesť webovú metódu, ktorá nesie v sebe obsah, že je to webová služba:

```
[WebMethod(Description = "Vkladam do XML suboru")]
```

*Ajaxom* sú dáta poslané do premennej *udaje*, kde je uložené heslo a dátum predčasného odpredaja bankovej zmenky. Parameter *udaje* je *string*, ktorý je treba rozdeliť,

aby bolo možné vyhľadávať. Aby sa dalo pracovať s XML súborom, je potrebné nastaviť cestu k xml súborom:

```
private string FileName =  
HttpContext.Current.Server.MapPath("~/App_Data/zakaznici.xml");
```

Pre otvorenie XML súboru v jazyku C# je použitie objektu doc triedy XmlDocument:

```
doc = new XmlDocument();
```

XML súbor je následné prečítaný a uložený do dátového prúdu, cez ktorý web služba pracuje s xml súborom:

```
doc.Load(FileName);
```

Každý xml dokument musí obsahovať koreňový element. Koreňovým elementom v xml vklady.xml je <zakaznici> </zakaznici>. Podľa hesla a dátumu, ktorý užívateľ zadal pri vkladaní do webového klienta, webová metóda vyhľadá v xml súboru tie detské elementy, v ktorých sa zhoduje heslo. Na prehľadávanie xml súboru, pre nájdenie zhody hesla s heslom, ktoré bolo zadané užívateľom vo webovom klientovi, je použitá XmlNodeList. NodeList vytvorí inšanciu triedy XmlNodeList.

```
XmlNodeList nodeList = docVklady.SelectNodes("/zakaznici/zakaznik");
```

Do premennej nodeList, ktorá je inštančnou triedy XmlNodeList, sú vložené všetky detské elementy rodičovského elementu *zakaznik*, ktorý je detským elementom, elementu *zakaznici*. Pre prehľadávanie jednotlivých vybraných detských elementov je použitý cyklus *foreach*, pomocou ktorého sa prehľadajú všetky detské elementy. Vyberie sa ten detský element, v ktorom sa nájde zhoda v elemente <heslo> s heslom zadaným zákazníkom. Na obrázku č. 4 je heslo užívateľom zadané LA. Cyklus *foreach* je spracovaný nasledovne:

```

foreach (XmlNode xn in nodeList)
{
    if (xn["heslo"].InnerText == sHeslo)
    {
        meno = xn["meno"].InnerText;
        priezvisko = xn["priezvisko"].InnerText;
        banka = xn["banka"].InnerText;
        suma = xn["suma"].InnerText;
        datum = xn["datum"].InnerText;
    }
}

```

V cykle cez *foreach* prehľadávame všetky detské elementy rodičovského elementu *zakaznik*, ktorý je detským elementom elementu *zakaznici* zo súboru *vkklady.xml* a porovnávame pomocou *if*, či sa zhoduje heslo v xml súbore s heslom, ktoré bolo zadané užívateľom. V prípade zhody uložíme všetky elementy (*meno*, *priezvisko*, *banka*, *suma*, *datum*) detského elementu *zakaznik*, do premenných *meno*, *priezvisko*, *banka*, *suma*, *datum*. Ak zhoda v heslách nenastane, cyklus pokračuje ďalej až po posledný detský element. Ak zhoda hesla v xml súbore s heslom zadaným užívateľom nastala, tak webová metóda vyberie nasledovné dáta:

```

<zakaznik>
  <meno>Lukas</meno>
  <priezvisko>Jancek</priezvisko>
  <banka>KONTROLA+0.09</banka>
  <suma>30000</suma>
  <datum>30.6.2013</datum>
  <heslo>LA</heslo>
</zakaznik>

```

Obr. 5 Detské elementy rodičovského elementu *zakaznik*

#### 4.2.2. Rátanie obchodného diskontu

Ak sa zhodujú uložené heslá a dáta, nastane vypočítavanie výšky obchodného diskontu. Ak užívateľ predčasne odpredá bankovú zmenku banke, tá si potom zrazí obchodný diskont pri ročnej diskontnej sadzbe, ktorú má každá banka definovanú.

Obchodný diskont sa vyráta zo vzorca: (Výška bankovej zmenky \* diskontná sadba \* (dátum splatnosti – dátum predčasného odpredaja))/360<sup>28</sup>.

V prípade predčasného odpredaja zákazníkom je potrebné vypočítať rozdiel v počte dní. Ak užívateľ zadal pri kúpe bankovej zmenky dátum 30.5.2013, ale bankovú zmenku plánuje predat' už skôr, dátumom 1.5.2013 tak rozdiel v počte dní je 29. Hodnoty v *xml* súbore sú uložené v stringoch, ale keďže webová metóda pracuje s desatinnými číslami, je potrebné konvertovať *string* na *float*. V jazyku C# je k dispozícii funkcia:

```
float Fv = float.Parse(suma);
```

Následne po konvertovaní *stringu* na *float* a vypočítaní obchodného diskontu je potrebné opäť konvertovať *float* na *string*, pre potreby uloženia údajov do XML súboru. Na konverziu *float* na *string* opäť jazyk C# ponúka funkciu:

```
string hodnota = Convert.ToString(float udaj);
```

#### 4.2.3. Uloženie údajov do XML súboru

Po tom, ako webová metóda vypočíta obchodný diskont a sumu, ktorú vyplatí zákazníkovi, sú údaje uložené do xml súboru s názvom *zakaznici.xml*. *Zakaznici.xml* má nasledovný tvar:

- meno: meno zákazníka
- priezvisko: priezvisko zákazníka
- banka: názov banky, v ktorej užívateľ vlastní bankovú zmenku

---

<sup>28</sup> PIRČ, V., GRINČOVÁ, A.: *Finančná matematika*. Košice: 2008, ISBN 978-80-8073-986-7.

- suma: výška vlastnenej bankovej zmenky
- pokuta: diskont, ktorý je zrazený z bankovej zmenky
- odpocitane: suma, ktorá bude vyplatená zákazníkovi

```
<zakaznik>
  <meno>Lukas</meno>
  <priezvisko>Jancek</priezvisko>
  <banka>KONTROLA</banka>
  <suma>30000</suma>
  <pokuta>232,5</pokuta>
  <odpocitane>29767,5</odpocitane>
</zakaznik>
```

Obr. 6 Vloženie výsledných údajov do xml súboru zakaznici.xml

Vloženie údajov do XML súboru prebieha nasledovne. Vytvoríme detský element *zakaznik* koreňového elementu *zakaznici*. V elemente *zakaznik*, ktorý je detským elementom *zakaznici* vytvoríme detský element *meno*, *priezvisko*, *banka*, *suma*, *pokuta*, *odpocitane*. Príklad uvádza vloženie detského elementu *meno*, rodičovského elementu *zakaznik*. Postupne sa pokračuje rovnako ako v príklade, len s inými údajmi. Hodnota *meno* sa nahrádza postupne *priezvisko*, *banka*, *suma*, *pokuta*, *odpocitane*.

```
XmlElement subRoot = doc.CreateElement("zakaznik");
XmlElement appendedElementMeno = doc.CreateElement("meno");
XmlText xmlTextMeno = doc.CreateTextNode(meno);
appendedElementMeno.AppendChild(xmlTextMeno);
subRoot.AppendChild(appendedElementMeno);
doc.DocumentElement.AppendChild(subRoot);
doc.Save(FileName);
```

#### 4.2.4. Vrátenie transformovaných dát do webového klienta

Po úspešnom vložení údajov do XML súboru je potrebné, aby sa užívateľovi v klientskom okne webového prehliadača zobrazila informácia o správnom uložení dát. Po nájdení zhody hesla, ktoré bolo zadané užívateľom, a hesla, ktoré je uložené v xml súbore, sa užívateľovi pri následnom vypočítaní obchodného diskontu v klientskom okne zobrazia nasledovné informácie, ktoré informujú o sume bankovej zmenky, ktorú klient vlastnil, dátum výberu, obchodný diskont, ktorý mu bude zrazený, a suma, ktorá mu bude

vyplatená. Suma, ktorá bude vyplatená klientovi, je celková suma a od nej odrátaný obchodný diskont.

Nájdene údaje						
Meno	Priezvisko	Banka	Suma	Datum	Obchodný diskont	Vyplatené
Peter	Novak	KONTROLA	30000€	30.6.2015	232,5€	29767,5€

Obr. 7 Vypočítaný obchodný diskont

Výsledok, ktorý je zobrazený na obrázku, sú už dáta transformované pomocou *javascriptu* a vložené do klientskeho okna pomocou *ajaxu*. Ak by dáta neboli transformované, výsledok, ktorý by bol vrátený klientovi v podobe xml formátu by bol nasledovný:

<string>Peter</string>

<string>Novak</string>

<string>KONTROLA</string>

<string>3000€</string>

<string>30.6.2015</string>

<string>232.5€</string>

<string>29767.5€</string>

V tomto prípade výstup, ktorý bol vrátený webovou službou do klientskeho okna, nie je zrozumiteľne čitateľný pre osobu, ktorá nemá informatické vedomosti. Preto je použitá transformácia vráteného *xml* tvaru do tvaru *html*. Dáta, ktoré chceme vrátiť klientovi do prehliadača najskôr zreťazíme do *stringu* a následne celý reťazec pošleme do *javascriptu*. V jazyku C# je možnosť použitia *IDictionary*, ktorý reprezentuje generickú kolekciu kľúčov a hodnôt. V nasledovnom fragmente kódu sú obe hodnoty typu *string*:

```
IDictionary<string, string> data = new Dictionary<string, string>();
```

Prvý parameter je *key*:

- Key: reprezentuje typ v slovníku.

Druhým parametrom je *value* (hodnota):

- Value: reprezentuje typ hodnoty v slovníku.

Pre pridávanie do slovníka *IDictionary* do premennej *data* sa budú vkladať jednotlivé elementy, ktoré boli vybrané z xml súboru, a ktoré budú vrátené užívateľovi do prehliadača. Zápis pre pridanie hodnoty do slovníka je nasledovný:

```
data["meno"] = meno;
```

V nasledovnom príklade je kľúč key : “meno“ a value hodnota je meno. Pre všetky ostatné hodnoty je zápis rovnaký. Následne je potrebné slovník serializovať pomocou *JavaScriptSerializer*.

*JavaScriptSerializer* vykonáva serializáciu a deserializáciu pre AJAX-ovo podporované aplikácie.

Slovník data je vrátený do AJAX-u pomocou serializátora:

```
string strJSON = js.Serialize(data);  
  
return strJSON;
```

V Javacsiprite je definovaný AJAX, ktorý posiela dáta do súboru s príponou *axmx.cs*, a rovnako aj prijíma dáta. AJAX je štandardne definovaný v súbore s príponou *.js*. Z webovej služby sú dáta vrátené do AJAX-u v tvare *json reťazca*.

```
{ "meno": "Peter", "priezvisko": "Novak", "banka": "KONTROLA", "suma": "30000", "datum": "30.6.2013", "pokuta": "225", "rozdiel": "29775" }
```

Pre rozparovanie tohto *json reťazca* nám slúži z *jquery knižnice* funkcia *\$.parseJSON*.

```
$("#inserted_query").empty().append('<table>' +  
    '<tr class="first"><th colspan=8>Nájdené údaje</th></tr>' +  
    '<th>Meno      </th><th>Priezvisko    </th><th>Banka      </th><th>Suma</th><th>Datum</th><th>Pokuta</th><th>Vyplat</th>' +  
    '<tr>' +  
    '<td>' + data.meno + '</td>' +  
    '<td>' + data.priezvisko + '</td>' +  
    '<td>' + data.banka + '</td>' +  
    '<td>' + data.suma + '&euro;</td>' +  
    '<td>' + data.datum + '</td>' +  
    '<td>' + data.pokuta + '&euro;</td>' +  
    '<td>' + data.rozdiel + '&euro;</td>' +  
    '</tr>' +  
    '</table>');
```

### 4.3. Pridanie novej banky

Užívateľ, ktorý vlastní bankovú zmenku, si vyberá banku a jej diskontnú sadzbu, v ktorej si zakúpil zmenku. Webový prehliadač ponúka možnosť pridávania nových bánk a diskontných sadzieb. Po úspešnom pridaní novej banky a jej diskontnej sadzby je automaticky zobrazená aj v rozbaľovacom okne. Po spustení webovej služby webový klient ponúkne formulár s dvoma textovými poliami, pre vloženie nového názvu banky a pre vloženie diskontnej sadzby.





Vloženie banky a diskontnej sadzby

Názov banky

Diskontná sadzba


Pridaj Banku

Obr. 8 Vloženie banky a jej diskontnej sadzby

Na tlačidle *Pridaj Banku* je vytvorená udalosť, ktorá je vyvolaná po kliknutí. Po tom, ako užívateľ klikol na tlačidlo *Pridaj Banku* sa dáta serializovali. Serializácia bola vykonaná pomocou jQuery funkcie:

```
var data = $("#main_form_banku").serialize();
```

Premenná nachádzajúca sa v zátvorkách, *#main\_form\_banku*, je identifikátor formulára, pre vloženie údajov do xml súboru *banka.xml*. Dáta, ktoré boli užívateľom vyplnené, sa serializovali (vytvorí sa textový reťazec), a ten je uložený do premennej *data* a následne je poslaný webovej službe, ktorá je uložená v súbore s názvom *WebService\_XML\_file.asmc.cs*. Pri vkladaní novej banky sú dáta uskladnené do xml súboru s názvom *banka.xml*. Štruktúra xml súboru *banka.xml* je reprezentovaná koreňovým elementom `<banky><banky>`, v ňom je definovaný rodičovský element `<banka></banka>` a v rodičovskom elemente sú definované detské elementy `<nazov></nazov>` a `<diskontnasadzba></diskontnasadzba>`.



Názov banky

TOPTAS

Diskontná sadzba

0.85

Pridaj Banku

Obr. 9 Užívateľom vloženie banky a jej diskontnej sadzby

Pre ukážku je do políčka Názov banky vpísaná banka *TOPTAS* s diskontnou sadzbou 0,85. Nová banka a jej diskontná sadzba je uložená v xml súbore *banka.xml* nasledovne:

```
<banka>
  <nazov>TOPTAS</nazov>
  <diskontnasadzba>0.85</diskontnasadzba>
</banka>
```

Obr. 10 XML štruktúra pre vloženie banky

Pre úspešné vloženie do súboru *banka.xml* je potrebné súbor najskôr otvoriť, zapísať do neho údaje na koniec a napokon súbor zatvoriť. Cesta k súboru *banka.xml* je relatívna:

```
private string FileNameBanka =
HttpContext.Current.Server.MapPath("~/App_Data/banka.xml");
```

Do premennej *FileNameBanka*, ktorá je typu string, je vložená cesta ku xml súboru *banka.xml*.

```
private XmlDocument docBanka;
```

Premenná *docBanka* je typu *XmlDocument* a do nej bude vložený obsah xml súboru *banka.xml*, ktorý bude načítaný pomocou:

```
docBanka.Load(FileNameBanka);
```

Funkcia, ktorá vykonáva vkladanie novej banky do xml súboru *banka.xml*, je:

```
public string PridajBanku(string udaje)
```

Vstupným parametrom funkcie *PridajBanku* je string *udaje*. V reťazci *udaje*, ktoré sú poslané z javascriptu ajaxom sú json reťazcom uložené údaje názov novej banky a hodnota diskontnej sadzby banky.

```
{"d":{"banka\":"ff\","sadzba\":"2\"}}
```

Dáta, ktoré užívateľ zadal do textových polí, boli serializované do nasledovného json tvaru a poslané do webovej služby a uložené do premennej *udaje*, ktorý je vstupným parametrom funkcie *PridajBanku*. Z reťazca *udaje* je potrebné vybrať hodnoty *ff* a *2*

a uložiť ich do xml súboru *banka.xml*, následne vybrať vložené údaje z xml a dáta transformovať a poslať späť do klientskeho prehliadača. Vo webovom prehliadači sú dáta už transformované a čitateľné:

Následovná banka bola úspešne pridaná	
Banka	Diskontná sadzba
TOPTAS	0.85

Obr. 11 Vrátene transformované dáta z xml súboru po vložení novej banky

#### 4.4. Prehľadanie XML súboru banka.xml

Súbor *banka.xml* je načítaný do premennej *docBanka*. Následne je potrebné vyhľadať potomkov v xml súbore, ktorého rodičovským elementom je `<banky></banky>`.

```
XmlNodeList nodeList = docBanka.SelectNodes("/banky/banka");
```

Premenná *nodeList* je typu *XmlNodeList*, do ktorej sa ukladajú všetci potomkovia rodičovských elementov. Rodičovský element je `<banka></banka>`, ktorého koreňovým elementom je `<banky></banky>`. Premenná *docBanka*, v ktorej je už uložený celý obsah xml súboru *banka.xml* sa začne prehľadávať pomocou príkazu *SelectNodes*, ktorý je inštančnou metódou *docBanka*. Vstupným parametrom je koreňový element *banky* a vyhľadáva všetky detské elementy rodičovského elementu *banka*.

```
foreach (XmlNode xn in nodeList)
{
    if (xn["nazov"].InnerText == sBanka)
    {
        banka = xn["nazov"].InnerText;
        diskontnasadzba = xn["diskontnasadzba"].InnerText;
    }
}
```

Pomocou cyklu *foreach* budú prehľadávané všetky vybrané detské elementy rodičovského elementu `<banka>`. Pri nájdení zhody detského elementu `<nazov></nazov>` s hodnotou, ktorá bola zadaná užívateľom, pri vkladaní novej banky sa dáta vybrané z xml

súboru uložia do premenných *banka* a *diskontnasadzba*. Tie sa následne uložia do slovníka, ktorý je definovaný dvoma hodnotami string:

```
IDictionary<string, string> result = new Dictionary<string, string>();
```

Následne sú uložené hodnoty *banka* a *diskontnasadzba* do premenných:

```
result["banka"] = banka;  
result["sadzba"] = diskontnasadzba;
```

Premenná *result*, je následne serializovaná pomocou javascriptovského serializátora:

```
JavaScriptSerializer();
```

a vrátená späť do ajax-u kde je spracovávaný v podobe reťazca tvaru *json*. V ajaxe sú dáta uložené do premennej *response* a sú vrátene v json reťazci, ktorý je potrebné rozdeliť:

```
{"d":{"banka\":"ABC BANKA\","sadzba\":"0.17\"}}
```

Na rozdelenie reťazca ponúka jquery knižnička funkciu *parseJSON (parameter)* .

```
success: function (response){  
var data = $.parseJSON(response.d);  
}
```

Premenné, ktoré boli vrátené webovou službou a rozdelené pomocou jquery funkcie, sú uložené do premenných *data.banka*, *data.sadzba*. Nakoniec sú dáta naformátované a naštýlované do tabuľky a zobrazené v klientskom okne užívateľa. Pre kontrolu, či boli údaje vložené do xml súboru *banka.xml*, je jednoduché kliknúť v navigačnom paneli na ponuku **Vklad**, následne z rozbaľovacieho menu *Banka* si skontrolovať, či banka, ktorá bola vložená užívateľom, sa nachádza v rozbaľovacom menu.

**Vyplňte nasledovný formulár**

Meno

Priezvisko

Banka  
 OTH : 0.3€ ▼  
 OTH : 0.3€  
 UHBank : 0.8€  
 UKH Group : 0.74€  
 SSH : 0.09€  
 SVST : 0.07€  
 KONTROLA : 0.09€  
 posledna : 0.57€  
 Streda : 0.09€  
 AHA : 0.04€  
 DEXIA : 0.11€  
 TOPTAS : 0.85€

Obr. 12 Vloženie novej banky do súboru banka.xml

#### 4.5. Celkový sumár odpredaných bankových zmeniek

Celkový sumár odpredaných bankových zmeniek zobrazuje informácie o každom užívateľovi, ktorý si odpredal akúkoľvek bankovú zmenku v ktorejkoľvek banke. Klient po otvorení webového prehliadača a po kliknutí na možnosť z navigačného panelu *Zoznam* má možnosť usporiadania údajov vzostupne alebo zostupne podľa vlastného výberu:

- Bankovej zmenky
- Obchodného diskontu

Užívateľ si môže vybrať jednu zo štyroch možností usporiadania údajov. Po kliknutí na ľubovoľnú možnosť sa dáta začnú usporadúvať.



Obr. 13 Možnosť výberu usporiadania údajov

Ak užívateľ pri výbere bankovej zmenky uskutoční odpredaj, tak sa jeho transakcia uloží do xml súboru *zakaznici.xml*. Všetky transakcie sú ukladané do súboru a zobrazené do webového prehliadača. Po kliknutí v prehliadači na menu *Zoznam* sú dáta zobrazené v poradí ako boli uskutočňované odpredaje bankových zmeniek.

Posledné vložené údaje					
Meno	Priezvisko	Banka	Zmenka	Diskont	Odpočítaná suma
Peter	Novak	KONTROLA	30000€	225€	29775€
Karol	Repka	UHBank	10522€	4933,649€	5588,351€
Igor	Mrkva	OFS Nutrion	4000€	601,6667€	3398,333€

Obr. 14 Posledné uskutočnené predaje bankových zmeniek

#### 4.6. Usporiadanie bankovej zmenky zostupne

Po výbere možnosti usporiadania zmenky zostupne je ajaxom poslané do webovej služby, že sa žiada vyvolať funkciu, ktorá je definovaná:

```
public string UsporiadanieMax()
```

Funkcia *UsporiadanieMax()* pracuje so súborom *zakaznici.xml*, v ktorom sú uložené všetky vykonané transakcie vo výbere bankovej zmenky:

```
private string FileName =
HttpContext.Current.Server.MapPath("~/App_Data/zakaznici.xml");
```

Do premennej *FileName* je uložená relatívna cesta ku súboru *zakaznici.xml*. Následne v konštruktoze triedy je načítanie celého xml súboru do premennej *doc*. Súbor *zakaznici.xml* obsahuje koreňový element `<zakaznici></zakaznici>`, ktorého detským elementom sú `<zakaznik></zakaznik>`.

```
XmlNodeList nodeList = doc.SelectNodes("/zakaznici/zakaznik");
```

Pomocou inštančnej metódy *SelectNodes(parameter)* objektu *doc* vyberieme všetky detské elementy koreňového elementu *zakaznik*, ktorý obsahuje rodičovský element *zakaznici*. Pomocou príkazu

```
foreach (XmlNode xn in nodeList)
```

je prehľadávaný celý xml dokument *zakaznici.xml* a ukladany do dvojrozmerného poľa, v ktorom bude na každom indexe uložené meno, priezvisko, názov banky, výška vlastniacej bankovej zmenky, vypočítaný obchodný diskont a vyplatená suma. Do dvojrozmerného poľa budú ukladané hodnoty z xml súboru. Nech je dané pole *dat* nasledovne:

```
string[,] dat = new string[1000, 1000];
```

Pomocou príkazu *foreach* budú ukladané údaje do poľa s indexom nasledovne: Na začiatku je definovaný *i* ako index s hodnotou 0.

```
int i = 0;
```

Pomocou príkazu *foreach* je prehľadávaný celý xml súbor a hodnoty, ktoré budú nájdené v xml súbore sa budú ukladať do poľa nasledovne:

```
foreach (XmlNode xn in nodeList)
{
    dat[i, 0] = xn["meno"].InnerText;
    dat[i, 1] = xn["priezvisko"].InnerText;
    dat[i, 2] = xn["banka"].InnerText;
    dat[i, 3] = xn["suma"].InnerText;
    dat[i, 4] = xn["pokuta"].InnerText;
    dat[i, 5] = xn["odpocitane"].InnerText;
    i++;
}
```

Cyklus začína nájdením prvého detského elementu rodičovského elementu *<zakaznik></zakaznik>*. Hodnota premennej *i* je nastavená na 0. Na pozíciu *dat [0,0]* je uložené meno prvého detského elementu. Do *dat[0,1]* je uložené priezvisko, *dat[0,2]* je uložená banka, *dat[0,3]* je uložená výška kúpenej bankovej zmenky, *dat[0,4]* je uložený diskont, *dat[0,5]* je uložená celková suma po odrátaní diskontu od bankovej zmenky. Hodnota premennej *i* po dokončení prvého cyklu je navýšená na hodnotu 1 a následne sa nové dáta budú ukladať na pozície *dat[1,0]*, *dat[1,1]* až kým nebude prehľadávaný celý xml súbor a všetky dáta nebudú uložené do poľa *dat*. Aby bolo možné usporadúvať pole *dat* je

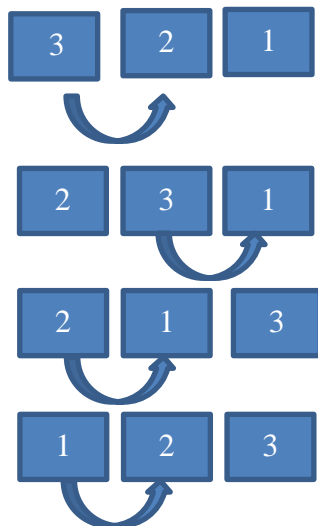
potrebné konvertovať hodnoty. Hodnoty v poli *dat* sú typu string a pre porovnávanie dvoch susedných prvkov a následnom upravovaní poradia je potrebné hodnoty konvertovať na tvar *int*. Konverziu v jazyku C# je možné vykonať príkazom:

```
int premenna = Convert.ToInt32(strHodnota);
```

*strHodnota* musí byť typu string. Ak sú už všetky údaje uložené v poli *dat*, je potrebné vykonať samotné bublinkové triedenie.

#### 4.6.1. Bublinkové triedenie (*Bubble sort*)

Bublinkové triedenie je často známe pod názvom *buble sort*. Je to jeden z najjednoduchších triediacich algoritmov, zo všetkých algoritmov, ktoré sa využívajú na triedenie. Z praktického hľadiska tento algoritmus nie je efektívny, pretože pri veľkom počte prvkov je výpočtový čas veľmi dlhý, pretože *buble sort* pracuje na princípe vymieňania susedných prvkov. Princíp *buble sortu* spočíva v prehľadávaní všetkých prvkov a vymieňania vždy prvkov ak je splnená podmienka, ktorá je zadaná užívateľom. Môže byť zoradenie prvkov vzostupne alebo zostupne a následkom skracovaní vždy o jeden krok v počte opakovaní.



Nech sú dané tri čísla v poradí zobrazenom na obrázku 2, 3, 1. *Bubble sort* začína prvým cyklom o dĺžke veľkosti poľa čiže 3. Druhý cyklus sa skracuje vždy o jeden krok. *Bubble sort* začne prehľadávať a porovnávať vždy susedné dva prvky. Ak je prvý prvok



väčší ako druhý, tak sú prvky vzájomne vymenené a najväčší prvok sa dostane na koniec zoznamu. Týmto spôsobom sa presúvajú prvky až po ukončení cyklu.

Nasledovný fragment kódu je preusporiadanie zoznamu pomocou bublinkového triedenia. Prvý cyklus začína prehľadávaním od 0 po *pocet*. Premenná *pocet* znamená počet rodičovských elementov v xml súbore *zakaznici.xml*. Podľa obrázka *Obr.15* je počet rodičovských elementov 4. Do premennej *pocet* je uložené číslo 4.

Posledné vložené údaje					
Meno	Priezvisko	Banka	Zmenka	Diskont	Odpočítaná suma
Peter	Novak	KONTROLA	30000€	225€	29775€
Karol	Repka	UHBank	10522€	4933,649€	5588,351€
Igor	Mrkva	OFS Nutrion	4000€	601,6667€	3398,333€
Tomas	Cibula	VSD ALL	30000€	7,5€	29992,5€

Obr. 15 Aktuálny stav posledných uskutočnených transakcií

Cyklus začína prvotným prehľadávaním podľa začínajúcim indexom 0 a končiacim pokiaľ nie je splnená podmienka. Premenná *pocet* je 4. Cyklus začína s hodnotou v premennej *j=0*. Pri prvom opakovaní je splnená podmienka, že  $j < 4$ . Vnorený cyklus, taktiež ako vonkajší, začína hodnotou pri premennej  $k = 0$  a rovnako aj počtom opakovaní to je 4 krát. Porovnávané hodnoty prekonvertujeme z typu *string* na typ *integer* a následne sú porovnané. Premenná *fr* (first), *sec* (second) sú porovnávané. Ak platí podmienka, že premenná *fr* je menšia ako *sec*, tak sa musia všetky hodnoty v poli vymeniť. Výmena je zabezpečená cyklom po podmienke:

```

for (int j = 0; j < pocet; j++)
{
    for (int k = 0; k < pocet; k++)
    {
        fr = Convert.ToInt32(dat[j, 3]);
        sec = Convert.ToInt32(dat[k, 3]);
        if (fr > sec)
        {
            for (int z = 0; z < 6; z++)
            {
                pom[j, z] = dat[j, z];
                dat[j, z] = dat[k, z];
                dat[k, z] = pom[j, z];
            }
        }
    }
}


```

Po skončení vonkajšieho cyklu je pole *dat* uložené do premenných, ktoré sú spolu zreťazené, uložené do slovníka a následne poslané ako výsledok do *ajaxu*. Ajax prijme dáta, ktoré sú už usporiadané a pošle klientovi tabuľku s usporiadanými údajmi:

Posledné vložené údaje					
Meno	Priezvisko	Banka	Zmenka	Diskont	Odpočítaná suma
Peter	Novak	KONTROLA	30000€	225€	29775€
Tomas	Cibula	VSD ALL	30000€	7,5€	29992,5€
Karol	Repka	UHBank	10522€	4933,649€	5588,351€
Igor	Mrkva	OFS Nutrion	4000€	601,6667€	3398,333€

Obr. 16 Usporiadanie bankových zmeniek zostupne

## 4.7. Usporiadanie bankovej zmenky vzostupne

Usporiadanie bankových zmeniek vzostupne je usporiadanie od najmenšiu po najvyššiu hodnotu. Rovnako ako pri usporiadaní zostupne od najvyššej hodnoty po najnižšiu sme použili rovnaký spôsob usporiadania, obmena nastane až pri podmienke v bublinkovom triedení. Po kliknutí na obrázok  sa vyvolá pomocou javascriptu akcia, že bolo kliknuté na tlačidlo a pomocou ajaxu je odoslaná udalosť webovej služby, aby načítala dáta z xml súboru *zakaznici*, dáta následne usporiada

vzostupne a vráti späť do klientskeho okna užívateľa. Vo webovej službe pre usporiadanie údajov od najnižšej po najvyššiu je definovaná funkcia:

```
public string UsporiadanieMin(){}

```

Definujeme si slovník, ako sme definovali v každej funkcii, pri ktorej sme vracali výstup užívateľovi.

```
IDictionary<string, string> result = new Dictionary<string, string>();

```

Na prehľadávanie xml súboru je využitý príkaz:

```
XmlNodeList nodeList = doc.SelectNodes("/zakaznici/zakaznik");

```

Do premennej doc, ktorá je typu *XmlDocument*, je uložený obsah xml súboru *zakaznici*. *SelectNodes* začne prehľadávať všetky detské elementy rodičovského elementu *zakaznik*, ktorý je koreňovým elementom, elementu *zakaznici*. Užívateľ po kliknutí v navigačnom menu *Zoznam* dostane na výstup neusporiadané údaje. Údaje sú zobrazené v poradí, ako boli vkladané do súboru. Banková zmenka, ktorá bola vybraná ako prvá, je v zozname na prvom mieste. Na poslednom mieste sa nachádza tá banková zmenka, ktorá bola vybraná ako posledná.

Posledné vložené údaje					
Meno	Priezvisko	Banka	Zmenka	Diskont	Odpočítaná suma
Peter	Novak	KONTROLA	30000€	225€	29775€
Karol	Repka	UHBank	10522€	4933,649€	5588,351€
Igor	Mrkva	OFS Nutrion	4000€	601,6667€	3398,333€
Tomas	Cibula	VSD ALL	30000€	7,5€	29992,5€

Obr. 17 Neusporiadaný zoznam bankových zmeniek

```
foreach (XmlNode xn in nodeList)
{
    dat[i, 0] = xn["meno"].InnerText;
    dat[i, 1] = xn["priezvisko"].InnerText;
    dat[i, 2] = xn["banka"].InnerText;
    dat[i, 3] = xn["suma"].InnerText;
    dat[i, 4] = xn["pokuta"].InnerText;
    dat[i, 5] = xn["odpocitane"].InnerText;
    i++;
}

```

Po tom, ako sú vyhľadane všetky detské elementy rodičovského elementu *zakaznici*, je prehľadaný každý detský elemnt a uložený do slovníka po indexoch. Následne je použité bublinkové prehľadávanie:

```
for (int j = 0; j < pocet; j++)
{
    for (int k = 0; k < pocet; k++)
    {
        fr = Convert.ToInt32(dat[j, 3]);
        sec = Convert.ToInt32(dat[k, 3]);
        if (fr < sec)
        {
            for (int z = 0; z < 6; z++)
            {
                pom[j, z] = dat[j, z];
                dat[j, z] = dat[k, z];
                dat[k, z] = pom[j, z];
            }
        }
    }
}
```

Vždy aktuálne porovnávané dva susedné prvky sú konvertované z typu *string* na typ *float*. Zmena oproti usporiadaniu vzostupne je len v podmienke. Kým pri usporiadaní zostupne je podmienka postavená *fr > sec*, pri usporiadaní vzostupne je to opačne *fr < sec*. Po tom, ako platí podmienka, že predchádzajúci prvok je menší ako nasledujúci, vykonáme vzájomnú výmenu prvkov. Pri výmene je potrebné použiť pomocné pole. Pretože nasledujúci prvok nahradí predchádzajúci, preto predchádzajúci prvok uložíme do pomocnej premennej, následne je nahradený predchádzajúci prvok nasledovným a nasledovný je prepísaný prvkom, ktorý je uložený v pomocnom poli. Po skončení cyklu bublinkového triedenia je pole serializované a poslané do ajaxu, ktorý prijme pole vo funkcii s parametrom *response*, kde je uložený celý výstup, ktorý sa má zobrazit' na obrazovku.

Posledné vložené údaje					
Meno	Priezvisko	Banka	Zmenka	Diskont	Odpočítaná suma
Igor	Mrkva	OFS Nutrion	4000€	601,6667€	3398,333€
Karol	Repka	UHBank	10522€	4933,649€	5588,351€
Peter	Novak	KONTROLA	30000€	225€	29775€
Tomas	Cibula	VSD ALL	30000€	7,5€	29992,5€


Obr. 18 Usporiadanie bankovej zmenky vzostupne

#### 4.8. Usporiadanie podľa diskontu zostupne

Usporiadanie zoznamu podľa diskontu spočíva v rovnakom princípe ako usporiadanie podľa výšky bankovej zmenky. Rovnako vo všetkých prípadoch sa začína s poľom ktoré je neusporiadané:

Posledné vložené údaje					
Meno	Priezvisko	Banka	Zmenka	Diskont	Odpočítaná suma
Peter	Novak	KONTROLA	30000€	225€	29775€
Karol	Repka	UHBank	10522€	4933,649€	5588,351€
Igor	Mrkva	OFS Nutrion	4000€	601,6667€	3398,333€
Tomas	Cibula	VSD ALL	30000€	7,5€	29992,5€

Obr. 19 Neusporiadaný zoznam odkúpených bankových zmeniek bankou

Po kliknutí na obrázok  je pomocou *javascriptu* vykonaná udalosť, ktorá pošle pomocou *ajaxu* do funkcie webovej služby:

```
public string PokutaMin(){}

```

signál o tom, že dáta majú byť usporiadané podľa výšky diskontu od najnižšieho po najvyšší. Keď webová služba zachytí informáciu od ajaxu o tom, že sa má vykonať akcia o usporiadaní, začne webová služba najskôr dáta načítat' z príslušného xml súboru

*zakaznici.xml*. Postup je rovnaký ako pri usporiadaní zostupne, vzostupne bankovej zmenky. Načítajú sa všetky detské elementy rodičovského elementu a uložia sa do poľa. Pole je následne vrátené do ajaxu, kde je už len naformátované a následne zobrazené klientovi vo webovom prehliadači:

Posledné vložené údaje					
Meno	Priezvisko	Banka	Zmenka	Diskont	Odpočítaná suma
Karol	Repka	UHBank	10522€	4933,649€	5588,351€
Igor	Mrkva	OFS Nutron	4000€	601,6667€	3398,333€
Peter	Novak	KONTROLA	30000€	225€	29775€
Tomas	Cibula	VSD ALL	30000€	7,5€	29992,5€

Obr. 20 Usporiadanie zoznamu podľa diskontu zostupne

#### 4.9. Usporiadanie podľa diskontu vzostupne

Poslednou alternatívou, ktorú si môže klient zvoliť, je usporiadanie výšky diskontu vzostupne od najnižšieho po najvyšší. Rovnako, ako vo všetkých prípadoch, aj v poslednom začína zoznam neusporiadane. Funkcia webovej služby

```
public string PokutaMax(){}
```

vykonáva usporiadanie zoznamu od najnižšej hodnoty po najvyššiu. Princíp usporiadania je taktiež rovnaký ako v prípadoch, ktoré sme uviedli vyššie *bublinkové triedenie*. Webová služba vráti do ajaxu pole, ktoré je už usporiadané, a v ajaxe sú už len naformátované do tabuliek a zobrazené vo webovom prehliadači.

Posledné vložené údaje					
Meno	Priezvisko	Banka	Zmenka	Diskont	Odpočítaná suma
Tomas	Cibula	VSD ALL	30000€	7,5€	29992,5€
Peter	Novak	KONTROLA	30000€	225€	29775€
Igor	Mrkva	OFS Nutrion	4000€	601,6667€	3398,333€
Karol	Repka	UHBank	10522€	4933,649€	5588,351€

Obr. 21 Usporiadanie podľa diskontu vzostupne

## Záver

Cieľom diplomovej práce bolo vytvorenie webovej služby, ktorá bude informovať o bankových zmenkách a ich diskontovaní. Webová služba je vytvorená pomocou *ASP.NET Framework 3.5* v jazyku *C#*. Ďalšie použité techniky, knižnice a jazyky sú *html*, *css*, *jquery*, *ajax*, *javascript*. Po otvorení webového klienta je užívateľovi navrhnuté zakúpenie bankovej zmenky, výber bankovej zmenky, vloženie novej banky do systému a celkový prehľad uskutočnených výberov bankových zmeniek. Vo webovom prehliadači sa vždy po kliknutí na *button* odchyť klik, uložia sa obsahy textových polí do premenných a tie sú následne poslané pomocou *ajaxu* webovej službe. Webová služba dáta uloží do jedného z troch xml súborov, podľa toho v akom okne sa užívateľ nachádza. Dáta, ktoré sú uložené do xml súboru, sú následne vybraté a vrátené do *javascriptu* v *json reťazci*, následne sú uložené do tabuľky a zobrazené klientovi ako výsledok. Priamo vo webovom prehliadači je možnosť si bližšie prečítať informácie o webovej službe, autorovi, použitých technikách. Webová služba pozostáva z jedného súboru typu *asmx* a štyroch *html* súborov. Každý jednej podstránke vo webovom prehliadači prilieha jeden *html* súbor. A všetky *html* súbory, ktoré sú na stránke, posielajú a prijímajú dáta do jedného súboru *asmx*, v ktorom je uložené jadro webovej služby. To následne pracuje s *xml súbormi*, ktoré sú uložené u klienta. Webová služba nie je rozdelená na dve časti, ako bývajú všetky webové služby, ktoré poskytujú banky a to *front end* a *back end*. Front end je tá časť stránky, ktorá sa zobrazuje len klientovi, ktorý si otvoril stránku. *Back end* ináč nazývaný aj *adminom*, je naopak tá časť, ku ktorej klient nemá prístup. V *back end* časti sú vykonávané všetky operácie a ovládanie stránky, ku ktorým má prístup len administrátor stránky. Aby si klient pri vkladaní mohol vybrať len tú bankovú zmenku, ktorú on vložil, tak pri vkladaní zadá heslo. Heslo je ľubovoľné a je priradené klientovi ku jeho bankovej zmenke. Týmto spôsobom je ošetrované, aby klient nemohol vyberať cudzie bankové zmenky. Heslo si klient zapamätá aby pri výbere použil jeho heslo, ináč mu bankovú zmenku webová služba nenájde. Heslo musí byť minimálne šesťznakové a musí obsahovať aspoň jedno veľké písmeno a aspoň jednu číslovku. Klient má možnosť si prezrieť všetky vykonané výbery bankových zmeniek. Následne má možnosť usporiadania už uskutočnených odkúpení bankových zmeniek, a to buď podľa výšky bankovej zmenky, alebo podľa diskontu vzostupne alebo zostupne.



## Zoznam použitej literatúry

PIRČ, V., GRINČOVÁ, A.: *Finančná matematika*. Košice 2008, ISBN 978-80-8073-986-7

SHOPcentrik. 2013. *ASP.NET – technológie*. [citované 24. február 2013]. Dostupné na: <<http://www.shopcentrik.sk/slovník/asp-net-technologie-3.aspx>>

MSDN. *Spoločný jazykový modul runtime (CLR)*. 2013. [online]. [citované 11. marec 2013]. Dostupné na: <<http://msdn.microsoft.com/cs-cz/library/8bs2ecf4.aspx>>

Wikipedia. 2013 In: *XML*. [online]. [2013-03-09]. [citované 10. marec 2013]. Dostupné na: <<http://sk.wikipedia.org/w/index.php?title=XML&action=history>>

Wikipedia. 2013. In: *ASP.NET*. [online]. [2013-03-15]. [citované 16. marec 2013]. Dostupné na: <<http://sk.wikipedia.org/wiki/ASP.NET>>

ANDRÁSSY, J.: *Serializácia štruktúrovaných dát*. [online]. [2007-11-14]. [citované 26. marec 2013]. Dostupné na: <<http://www.vsl.sk/vyuka/java07/XML.pdf>>

*Banková pôžička na zmenku*. 2009. [online]. [citované 2. apríl 2013]. Dostupné na: <<http://bankova-pozicka.webnode.sk/bankove-pozicka-na-zmenku/>>

*Pôžičky na zmenku ihneď a bez registra*. [online]. [citované 2. apríl 2013]. Dostupné na: <<http://uveryapozicky.sk/pozicky-na-zmenku-ihned-a-bez-registra>>

JANOVSKÝ, D.: *Čo je to Javascript*. [online]. Jak psat web. [2012-12-06]. [citované 3. apríl 2013]. Dostupné na: <<http://www.jakpsatweb.cz/javascript/javascript-uvod.html>>

LACKO, Ľ.: *Výhody a nevýhody technológie AJAX*. [online]. Infoware. [2009-12-07]. [citované 10. apríl 2013]. Dostupné na: <<http://www.itnews.sk/tituly/infoware/2009-12-07/c130666--vyhody-a-nevyhody-technologie-ajax>>

BĚHÁLEK, M.: *Jmenný prostor*. 2007. [online]. [citované 14. apríl 2013]. Dostupné na: <<http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/ch04s02.html>>

Microsoft Support. 2013. *Vymedzenie a vysvetlenie. Súbor DLL*. [online]. [citované 12. marec 2013]. Dostupné na: <<http://support.microsoft.com/kb/87934/sk>>

BÚR, V.: *Programovanie v jazyku Java*. 2008. [online]. [citované 2. apríl 2013]. Dostupné na: <<http://www.smnd.sk/viliam/Java.pdf>>

KOSTELNÍK, P.: *Java-XML::how know. Transformácia XML pomocou XSLT*. 2009. [online] [citované 22. marec 2013]. Dostupné na: <<http://neuron.tuke.sk/kostelni/tutorials/xml/xslt-transformation.html>>

MATĚNA, R.: *XML a značkovací jazyky, historie a vznik*. [online]. Web frequently asked questions. [2008-06-19]. [citované 10. marec 2013]. Dostupné na <<http://www.webfaq.cz/clanek/XML-a-znackovaci-jazyky-historie-a-vznik>>