

PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

DIPLOMOVÁ PRÁCE

WEBGRAPHS

Generátor grafů matematických funkcí jako server control v
asp.net



2007

Michal Horák

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně.

15. duben 2007

Michal Horák

Anotace

Cílem této bakalářské práce bylo vytvořit serverový ovládací prvek pro ASP.NET, který bude generovat grafy matematických funkcí ve 2D z analytického zadání a podporovat témata a datové vazby.

Děkuji panu doktoru Skoupilovi, za vedení této diplomové práce, za pomoc při vyskytnutých problémech a za určitou volnost při práci.

Obsah

1. Analýza problematiky webového programování	1
1.1. Rozdíly mezi webovými a desktopovými aplikacemi	1
1.2. Statické webové stránky	2
1.3. Dynamické webové stránky	2
1.3.1. Na straně klienta	3
1.3.2. Na straně serveru	4
1.3.3. AJAX	5
2. Principy a mechanismy tvorby serverových ovládacích prvků	6
2.1. Vlastnosti serverového ovládacího prvku	6
2.2. Stav a postback	7
2.2.1. Stav zobrazení	7
2.2.2. Správa stavu	8
2.2.3. Data postbacku	8
2.2.4. Spouštění postbacku	8
2.3. Složené ovládací prvky	9
2.4. Šablony ovládacích prvků	9
2.5. Podpora stylů	10
2.6. Podpora datových vazeb	11
3. Popis a analýza WebGraphs	12
3.1. Cíl projektu	12
3.2. Seznam požadavků	12
3.3. Dodatečné funkčnosti	12
3.4. Zvolené technologie	12
3.5. Návrh řešení	12
4. Implementace WebGraphs	14
4.1. Atributy Serverového ovládacího prvku Webgraphs	14
4.2. Třídy Serverového ovládacího prvku	16
4.3. Realizace	18
5. Použití WebGraphs ve Visual Studiu 2005	19
5.1. Přidání do projektu	19
5.2. Možnosti použití	19
6. Webová aplikace demonstrující použití WebGraphs	22
Závěr	24
Reference	25

Seznam obrázků

1.	Diagram realizace generování grafu	18
2.	Serverový ovládací prvek webgraphs v návrhovém zobrazení) . . .	19
3.	Prázdný graf	20
4.	Graph collection editor)	20
5.	Graf funkcí $\sin(x)$ a $\cos(x)$	21
6.	Webová aplikace demonstrující použití Serverového ovládacího prvku	23

1. Analýza problematiky webového programování

1.1. Rozdíly mezi webovými a desktopovými aplikacemi

Webové aplikace mají na rozdíl od těch desktopových několik specifík. Prvním specifíkem je, že jde vždy o spojení mezi serverem a klientem, kde klient je počítač uživatele připojený k síti internet a serverem je počítač se speciálním softwarem, který hostuje webovou aplikaci. Klient a server komunikují přes přenosový protokol HTTP (Hypertext transfer protocol). Na klientovi je potřeba spustit webový prohlížeč, kam se napíše adresa aplikace, která se přes DNS (Domain Name Servis) přepíše na IP adresu serveru, kde webová aplikace běží. Dále se vytvoří HTTP požadavek na přístup k danému souboru, otevře se socket, většinou na portu 80, a požadavek se odešle. Server požadavek vyhodnotí a odešle odpověď. Pokud jde o nějakou webovou stránku, pak vrátí například kód html a webový prohlížeč zobrazí webovou stránku.

Druhým specifíkem webové aplikace je, že přenos přes protokol http neudrží stav aplikace. Po každém odeslání odpovědi na http požadavek se vše zapomene. Při další činnosti se tedy musí odeslat nový požadavek a server pošle novou stránku. Tohle může být problém u aplikací, které si mají něco zapamatovat. Jako příklad se může uvést internetový obchod a nákupní košík. Tento problém se dá řešit několika způsoby, data si můžete uložit jako:

- Parametry do adresy url – tedy do proměnné, kterou si server vezme z url adresy a přes metodu GET si ji přečte. Příklad adresy s parametry: `http://www.google.com/search?q=dotaz`, zde máme uloženu proměnnou `q` a její hodnota je řetěz "dotaz"
- Cookies – tedy do proměnné, která se uloží do webového prohlížeče a může se jí nastavit doba platnosti
- Skryté pole ve formuláři – tento způsob využívá například ASP.NET, kde je každá stránka dělaná jako jeden formulář a ve skrytých polích je uložen tzv. `viewstate`. Tedy data se po odeslání formuláře dostanou na server převážně pomocí metody POST.

Jakmile se data dostanou na server, pak se na něm může spustit nějaká aplikace či skript a server pak vrátí v http odpovědi požadovaný výsledek, většinou soubor psaný v jazyku html, xhtml, xml, nebo může vrátit i obrázek či jakýkoliv jiný soubor. Webová aplikace je tedy takovou kolekcí souborů, které se můžou a nemusí generovat.

Dalším specifikem je, že webová aplikace generující webovou stránku musí vracet zdrojový kód v html případně XHTML. Samozřejmě, že by měl být kód validní podle standardů W3C (World Wide Web Consortium). Některé chyby jsou webové prohlížeče schopné eliminovat, u některých se pak o to alespoň pokouší, ale ne vždy úspěšně.

1.2. Statické webové stránky

Pro programování statických stránek nám stačí osvojit si základní znalost HTML (případně XHTML), kaskádových stylů a několik doporučených postupů. Samozřejmě, že je potřeba vše psát jak syntakticky tak i sémanticky správně, ale není to nutností. Webové prohlížeče se snaží zobrazit i špatně napsaný dokument a tak se někdy chyby ztratí.

HTML

HTML (HyperText Markup Language) je značkovací jazyk pro hypertext vycházející z SGML (Metajazyk pro definici značkovacích jazyků), který je určen pro psaní webových stránek. Jazyk je charakterizován množinou značek (tzv. tagů), které se uzavírají do lomených závorek.

Kaskádové styly CSS

CSS (Cascading Style Sheets) je jazyk, který odděluje vzhled webových stránek od obsahu. Je použitelný pro jazyky HTML, XHTML a všechny jazyky XML. Problémem kaskádových stylů je fakt, že každý prohlížeč je interpretuje trochu jinak. Díky tomu je těžší napsat styly tak, aby stránka vypadala ve všech nejpoužívanějších webových prohlížečích stejně. Používají se na to různé techniky:

- hackování – speciální CSS konstrukce, které podporují jen některé prohlížeče a tak je možné pro dané prohlížeče psát css individuálně (viz. <http://css.interval.cz/clanky/css-hacky-prehled/>)
- Podmíněné komentáře – podmíněné komentáře podporuje pouze největší nepřítel pro kodéry webových stránek MS Internet Explorer (MSIE) verze 5 a vyšší. Díky tomu můžeme pro MSIE a všechny jeho verze napsat zvláštní css styl. Stejně tak můžeme zapisovat i kód v html.

1.3. Dynamické webové stránky

Dynamické webové stránky jsou ty stránky, které reagují na akce uživatele. Například umožní změnu vzhledu načtené stránky, nebo umožní komunikaci s webovými službami či serverem.

1.3.1. Na straně klienta

Dynamické stránky na straně klienta jsou především ty, které dynamicky mění svůj vzhled, reagují se zadanými vstupy, a to bez odeslání požadavku na server. Vše se tedy děje ve webovém prohlížeči.

JavaScript

Nejpoužívanějším jazykem, který se používá na straně klienta je JavaScript. Jde o multiplatformní objektový jazyk, jehož syntaxe vychází z jazyků C a Java.

VBScript

VBScript je odpovědí Microsoftu na JavaScript. Tento jazyk vychází z Visual Basicu a je stejně jako JavaScript určený pro skriptování na webových stránkách.

ActiveX

ActiveX jsou malé aplikace, které je možné přidat do svých webových stránek. Mají vymezený svůj prostor, ve kterém mohou vykreslovat výstupy, číst vstupy z klávesnice, reagovat na události a podobně. ActiveX prvky mívají problémy s bezpečností a je to dáno tím, že nejsou izolovány od operačního systému a tak mohou obsahovat nebezpečný kód.

Java aplety

Podobné jako ActiveX, jsou napsané v jazyku Java a ke svému běhu potřebují Java Virtual Machine. Oproti ActiveX jsou ale o dost bezpečnější, protože jsou jejich práva kontrolována tzv. security managerem.

1.3.2. Na straně serveru

Stejně jako u dynamických stránek na straně klienta, tak i dynamické stránky mění svůj vzhled, data a vlastnosti na základě zadaných parametrů, ale tentokrát již na straně serveru. Klientovi je tak zaslán pouze výsledek skriptů.

Common Gateway Interface

Common Gateway Interface nebo zkráceně CGI je protokol, který umožňuje komunikaci externích aplikací s webovým serverem. To nám umožňuje zaslat požadavek na webový server, který spustí nějakou aplikaci mimo server a vrátí její výsledek. CGI byla první technologie, která umožňovala dynamické generování webových stránek na straně serveru.

PHP

PHP (Hypertext Preprocessor) je interpretovaný skriptovací jazyk běžící na straně serveru, který se kombinuje s html značkami pro vygenerování html dokumentu. Syntaxe je podobná jazyku C, ale najdeme zde i prvky z Pearlu nebo Javy. Jedná se o velice oblíbený jazyk hlavně proto, že je relativně jednoduchý, intuitivní a stojí za ním rozsáhlá komunita vývojářů. Od páté verze je tento jazyk dokonce objektově orientovaným jazykem, ale stále to má své mouchy.

ASP

Podobně jako PHP je i ASP skriptovacím jazykem na straně serveru od společnosti Microsoft, který může být psán zároveň s html a využívá com komponent.

ASP.NET

ASP.NET je platforma postavená na .NET frameworku pro programování webových aplikací. Programování pod ASP.NET je založeno na tzv. Common Language Runtime (CLR). Díky tomu můžou programátoři psát pro ASP.NET ve všech jazycích, které CLR podporuje. Nejčastěji se využívá Visual Basic .NET nebo C#. Webové aplikace v ASP.NET jsou oproti ASP nebo i PHP předkompilovány a tak díky tomu, že se již nemusí interpretovat, jsou rychlejší. ASP.NET je také objektově orientované, což se o PHP do verze 5 ani ASP jazycích říci nedá a oproti PHP 5 je podpora objektů o dost větší.

JSP a Java Servlets

JSP (Java Server Pages) je technologie firmy Sun Microsystems pro vývoj aplikací na straně serveru, u které se používá jazyku Java. Kód se píše opět přímo do značek html. Zdrojový kód se přeloží na servlet a ten je potom na serveru spuštěn. Servlety jsou tedy komponenty napsané v jazyce Java, které běží na straně serveru.

1.3.3. AJAX

Technologie AJAX (Asynchronous JavaScript and XML) využívá dynamiky jak na straně klienta, tak na straně serveru. Jde o zasílání asynchronních požadavků a odpovědí mezi webovým prohlížečem a serverem převážně přes XML. K tomu je využíván objekt XMLHttpRequest, který do svého prohlížeče MSIE implementoval Microsoft. Využití AJAXu je především v tom, že můžeme tvořit interaktivní webové stránky, kde se může měnit obsah bez potřeby obnovení stránky. Všechno se děje na pozadí. AJAX je nyní velice populární a najdete jej převážně u moderních webových aplikací označovaných jako web 2.0.

2. Principy a mechanismy tvorby serverových ovládacích prvků

V ASP.NET se při programování webových stránek hojně používají tzv. Serverové ovládací prvky. Jsou to vlastně takové komponenty stránky, které mají svoje vlastnosti, události a generují část webové stránky. V ASP.NET existují kromě serverových ovládacích prvků tzv. Uživatelské ovládací prvky. Rozdíl mezi nimi je v tom, že serverový ovládací prvek se kompiluje do DLL assembly a umožňuje programátorsky generovat html kód. Je možné zdědit vzhled a chování z již napsaného ovládacího prvku a dál ho rozšiřovat. Serverový ovládací prvek je použitelný ve všech možných webových aplikacích a stačí ho napsat pouze jednou. Svůj serverový ovládací prvek, který má například hledat nějaké zboží, může jeho vlastník rozeslat svým partnerům a ti jej následně pouze přidají na svůj web a vše začne hned fungovat. Stejně tak je možné si serverový ovládací prvek přidat do toolboxu a jeho okamžitému využívání pak již nic nebrání. Další výhodou serverového ovládacího prvku je, že ho lze nechat zobrazit v návrhovém zobrazení ve Visual Studiu a tak je většinou ihned zřejmé, jak bude vypadat.

Serverový ovládací prvek je možno vytvořit tak, že se napíše třída odvozená z System.Web.UI.WebControls.WebControl. Třída WebControl obsahuje všechny vlastnosti a funkce, které mají všechny serverové ovládací prvky společné. Tuto napsanou třídu lze buď zkompileovat do DLL assembly nebo ji dát do adresáře AppCode v projektu, kde programátor chce serverový ovládací prvek použít. Zde je ukázka jednoduchosti vytvoření Serverového ovládacího prvku:

```
public class Example: WebControl{
    protected override void Render(HtmlTextWriter output){
        output.write("Hallo world!");
    }
}
```

V rodičovské třídě WebControl se nachází funkce Render(), která se spustí vždy, když se dostaví požadavek na serverový ovládací prvek. Programátor si tuto funkci přepíše, aby mohl generovat svůj vlastní kód. Funkce má jako parametr objekt HtmlTextWriter, který generuje HTML kód ovládacího prvku. V tomto případě pouze vypsals na výstup řetěz „Hallo world!“, ale stejně tak může vypsats cokoliv jiného.

2.1. Vlastnosti serverového ovládacího prvku

Jestliže chce mít programátor Serverový ovládací prvek, který reaguje na nějaký vstup, potom mu musí určit nějaké vlastnosti. Proto změni třídu Example, přidá do ní vlastnost Text, kterou bude chtít nechat vypsát na výstup.

```
public class Example: WebControl{

    private string text;
    public string Text{
        get{
            return text;
        }
        set{
            text = value;
        }
    }

    protected override void Render(HtmlTextWriter output){
        output.write(Text);
    }
}
```

2.2. Stav a postback

Protože protokol http nedokáže mezi požadavky udržet stav webové aplikace, musíme využít další metody. ASP.NET používá tzv. stav zobrazení a postback. Informace mezi požadavky si uloží do stavu zobrazení a při odeslání zpět na server odešle z webové stránky kolekci s daty z formuláře.

2.2.1. Stav zobrazení

V našem příkladu serverového ovládacího prvku je vlastnost Text a jestliže tato vlastnost bude programově změněna, potom se po dalším odeslání požadavku na server ztratí. Pokud si programátor bude chtít tuto změněnou vlastnost ponechat, pak ji bude muset uložit do stavu zobrazení:

```
public string Text{
    get{
        string text = (String)ViewState["Text"];
        return text;
    }
    set{
        ViewState["Text"] = value;
    }
}
```

}

Každý ovládací prvek má svůj vlastní stav zobrazení, a proto s ním nemůže manipulovat žádný jiný ovládací prvek a ani nemůže manipulovat se stavem zobrazení stránky. Když se podíváte do vygenerovaného zdrojového kódu aspx stránky, pak v něm najdete jedno skryté pole „_VIEWSTATE“, kde je uložena speciální stromová struktura stavu zobrazení všech ovládacích prvků včetně stavu zobrazení stránky. Proto by se do stavu zobrazení neměly ukládat velké objekty. Všechno se pak uloží do zdrojového html kódu stránky, jejíž velikost samozřejmě roste a je potřeba více času pro její stažení.

2.2.2. Správa stavu

Jako alternativu ke stavu zobrazení má ASP.NET 2.0 tzv. Správu stavu (Control State), která funguje stejně jako stav zobrazení, ovšem s tím rozdílem, že správa stavu není ovlivněna vlastností EnableViewState. Dokonce se ukládá do stejného skrytého pole. Pro použití správy stavu je potřeba přepsat metodu OnInit a to tak, že se v ní zavolá funkce Page.RegisterRequiresControlState(this). Data do správy stavu uloží funkce SaveControlState() a načte funkce LoadControlState().

2.2.3. Data postbacku

Data postbacku jsou určena například pro ovládací prvky input (ten generuje serverový ovládací prvek textbox), který se od ostatních liší tím, že je možné jeho prostřednictvím měnit data. Jakmile dojde k postbacku, tak se data z těchto prvků stanou součástí informací v kolekci ovládacích prvků a serverový ovládací prvek textbox musí tato data zjistit, aby mohl změnit svůj stav. Ke zjištění těchto dat je potřeba napsat rozhraní IPostBackDataHandler. Tím se ovládacímu prvku umožní, aby po postbacku získal potřebná data.

2.2.4. Spouštění postbacku

Napsáním výše zmíněného rozhraní IPostBackDataHandler bude mít programátor možnost získat odeslaná data z každého postbacku. Postback může spustit kliknutím na jakýkoliv ovládací prvek, kterému nastavíme parametr onClick na JavaScriptovou funkci _doPostBack(), která je automaticky přidána do každé aspx stránky.

2.3. Složené ovládací prvky

Jestliže bude chtít programátor vytvořit jeden serverový ovládací prvek, který bude využívat několik jiných již hotových ovládacích prvků, pak mu stačí vytvořit třídu, která bude dědit z třídy `System.Web.UI.WebControls.CompositeControl`, která je také zděděna z `WebControl`. Následně překryje metodu `CreateChildControls()` tak, že v ní dynamicky vytvoří požadované ovládací prvky a přidá je do vlastnosti `Controls`.

```
public class Example: CompositeControl{
    protected Label label;
    protected TextBox textBox;

    public Text{
        get{return (String)ViewState["Text"];}
        set{ViewState["Text"]=value;}
    }

    protected override void CreateChildControls(){
        label = new Label();
        label.Text = "Popisek";
        // vypne stav zobrazení
        label.EnableViewState = false;
        // Přidá ovládací prvek do složeného ovládacího prvku
        Control.Add(label);
        textBox = new TextBox();
        textBox.Text = Text;
        textBox.EnableViewState = false;
        Control.Add(textBox);
    }
}
```

2.4. Šablony ovládacích prvků

U složených ovládacích prvků bývá potřeba, aby bylo možné měnit jejich rozvržení a vzhled. K tomu by měly dopomoci šablony ovládacích prvků. Když se tedy vezme náš složený ovládací prvek, pak mu lze přidat vlastnost:

```
private ITemplate itemTemplate

public ITemplate ItemTemplate{
    get{return itemTemplate;}
    set{itemTemplate = value;}
}
```

Potom stačí pro každý ovládací prvek zavolat metodu `itemTemplate.InstantiateIn(control)`, kde `control` je objekt našeho ovládacího prvku. Takže příklad složeného ovládacího prvku by mohl vypadat následovně:

```
public class Example: CompositeControl{
    protected Label label;
    protected TextBox textBox;

    private ITemplate itemTemplate

    public ITemplate ItemTemplate{
        get{return itemTemplate;}
        set{itemTemplate = value;}
    }

    public Text{
        get{return (String)ViewState["Text"];}
        set{ViewState["Text"]=value;}
    }

    protected override void CreateChildControls(){
        label = new Label();
        label.Text = "Popisek";
        // vypne stav zobrazení
        label.EnableViewState = false;
        // Přidá ovládací prvek do složeného ovládacího prvku
        itemTemplate.InstantiateIn(label);
        Control.Add(label);
        textBox = new TextBox();
        textBox.Text = Text;
        textBox.EnableViewState = false;
        itemTemplate.InstantiateIn(textBox);
        Control.Add(textBox);
    }
}
```

Samozeřejmě, že takovýchto šablon je možné udělat pro jeden ovládací prvek více, třeba šablonu pro hlavičku, tělo a patičku. Pak se jen přidají jednotlivé ovládací prvky do správné šablony.

2.5. Podpora stylů

Jestliže je potřeba, aby bylo možné měnit vzhled serverového ovládacího prvku pomocí html, pak se musí přidat nějaké veřejné vlastnosti, kde se nastaví barvy,

fonty apod. a programově tyto vlastnosti přidat do generovaného html kódu. Samozřejmě, že již některé vlastnosti stylu jsou nastaveny přímo v třídě WebControl, ze které vychází všechny vlastní serverové ovládací prvky.

2.6. Podpora datových vazeb

Pokud je potřeba vytvořit serverový ovládací prvek s podporou datových vazeb, pak ho musí programátor dědit z System.Web.UI.WebControls.DataBoundControl a překrýt vlastnost DataSource a metodu DataBind.

```
private ArrayList seznam;

public override object DataSource
{
    get
    {
        object s = seznam;
        return s;
    }
    set
    {
        seznam = value;
    }
}

public override void DataBind()
{
    base.DataBind();

    // Další kód, který bude dále pracovat s vlastností DataSource
}
```

3. Popis a analýza WebGraphs

3.1. Cíl projektu

Cílem projektu je naprogramování serverového ovládacího prvku pro ASP.NET 2.0 generujícího grafy základních matematických funkcí v2D podporující datové vazby, skiny a styly. Dále vytvoření ukázkové aplikace demonstrující možnosti výše zmíněného ovládacího prvku.

3.2. Seznam požadavků

- Analýza problematiky webového programování.
- Popis principů, mechanismů tvorby a použití server controls
- Vytvoření server control pro dynamické vykreslování grafů matematických funkcí
- Podpora datových vazeb, stylů, skinů, apod.
- Webová aplikace demonstrující použití vytvořeného server controlu
- Jazyk C# nebo VB

3.3. Dodatečné funkčnosti

- Generování do formátů JPEG, BMP, GIF a PNG
- Možnost přidání více grafů do jednoho obrázku

3.4. Zvolené technologie

Serverový ovládací prvek pro ASP.NET 2.0 bude psán pomocí jazyka C# za pomoci knihovny GDI+.

3.5. Návrh řešení

Serverový ovládací prvek bude generovat grafy pomocí jazyka C# a knihovny GDI+, která je k dispozici přímo ve Visual Studiu 2005, ve kterém budu zadaný webový ovládací prvek implementovat. Tato knihovna dává k dispozici možnost generovat grafické objekty, se kterými lze následně libovolně manipulovat. Takový matematický graf lze vytvořit spojením několika vypočtených bodů křivkou nebo linkou. Počet bodů bude roven šířce generovaného grafu v pixelech. Pro výpočet funkčních hodnot analyticky zadané funkce byla

využita volně dostupná knihovna pro ASP .NET "The expression evaluator" [<http://www.codeproject.com/useritems/eval3.asp>], která z řetězce přečte názvy funkcí a proměnných a ze zadané třídy spustí odpovídající funkce vracující výsledek. V případě, že funkce je v nějakém bodě nedefinována, pak bude tento bod vynechán a u posledního spočítaného bodu uzavřeme křivku. Pokud je nutné pokračovat v generování grafu, potom začneme generovat křivku novou ze zbývajících bodů. Jakmile se graf v nějakém bodě dostane mimo horní nebo dolní okraj obrázku, pak budeme s tímto bodem zacházet stejně, jako by v něm nebyla funkce definována.

Serverový ovládací prvek generuje grafy do obrázkových souborů, které po určitou dobu nechává na serveru, aby se zbytečně neplýtvalo výpočetním výkonem. Pokud již byl graf jednou vygenerován, pak dokud nebude smazán, již znovu generován nebude a bude rovnou zobrazen. Název souboru se pak generuje ze zadaných parametrů ovládacího prvku, které se uloží do řetězce, který je následovně zahashován. Serverový ovládací prvek také podporuje témata, která představují pozadí pod grafem, šířku a barvu os a také šířku a barvu grafu. Témata si může každý sám doplnit tak, že přidá nový adresář do adresáře themes v kořenovém adresáři, kam přidá soubor theme.xml, kde je vše nedefinováno. Takový soubor může vypadat například takto:

```
<?xml version="1.0" encoding="utf-8" ?>
<theme background="background.jpg" padding="27">
  <axis view="true" size="1" color="Black" numbers="1"
    numbersOnAxis="true" font="Verdana"/>
  <graph color="Black" size="1" />
  <thumbnail file="example.gif" />
</theme>
```

Témata lze také změnit pomocí atributů. A pokud se bude generovat více grafů do jednoho obrázku, pak lze každý graf jednotlivě obarvit a také nastavit jeho šířku.

4. Implementace WebGraphs

Pro vytvoření serverového ovládacího prvku je potřeba napsat třídu, která se dědí z jedné ze tříd Control, WebControl nebo DataBoundControl. Protože je v požadavcích podpora datových vazeb, pak je nejlepší dědit z poslední jmenované třídy. Následně je potřeba do této třídy napsat všechny atributy, které mají být viditelné z venku a to pomocí get a set. Samozřejmě musejí být tyto atributy veřejné.

4.1. Atributy Serverového ovládacího prvku Webgraphs

Appearance

- Text (string) – nadpis grafu
- Description (string) – popis grafu

Data

- DataSource (object) - objekt, jemuž se přiřadí seznam bodů, který následně navážeme na ovládací prvek a spojením těchto bodů vytvoříme graf.

Graphs

- Graphs (collection) - kolekce objektů Graph
- ImageFormat (enum) - formát obrázku, který je výstupem ovládacího prvku
- Points (collection) - kolekce objektů GPoint

Layout

- Height (unit) - výška obrázku
- Weight (unit) - šířka obrázku

Theme

- Axis (bool) - zobrazit osy?
- AxisColor (System.Drawing.Color) - barva os
- AxisFont (string) - písmo čísel na osách
- AxisSize (int) - šířka os v pixelech

- GraphSize (int) - šířka grafu v pixelech
- ChangeTheme (bool) - změna téma podle zadaných parametrů
- Numbers (int) - počet čísel na každé ose
- NumbersOnAxis(bool) - zobrazit čísla na osách?
- Padding (int) - odsazení grafu od okrajů obrázku
- Theme (string) - název tématu

Jestliže je mezi atributy nějaká kolekce, potom je potřeba udělat atribut jako seznam objektů a povolit, aby měl tento atribut další vnitřní vlastnosti. Proto se vytvoří nová třída, jejíž instancí bude objekt obsahující další atributy, které je následně také možno zadat pro generování grafu. Protože má každý objekt kolekce další svoje atributy, tak je musím také popsat a tady jsou:

Collection Graphs

Axis

- From_x - minimální možná hodnota na ose x, která se má zobrazit
- To_x - maximální možná hodnota na ose x, která se má zobrazit
- From_y - minimální možná hodnota na ose y, která se má zobrazit
- To_y - maximální možná hodnota na ose y, která se má zobrazit

Graph

- Color - barva zadaného grafu, pokud bude atribut ChangeTheme nastaven na true, pak bude mít graf tuto barvu
- Function - funkce zadaného grafu, který se má generovat
- Size - šířka zadaného grafu, pokud bude atribut ChangeTheme nastaven na true, pak bude mít graf tuto šířku

Points

- Points - Kolekce bodů, které se mají v grafu vyznačit

Collection Points

- X - souřadnice x
- Y - souřadnice y

4.2. Třídy Serverového ovládacího prvku

Třída WebGraphsControl

Třída WebGraphsControl je třída pro inicializaci serverového ovládacího prvku

Metoda	Popis
DataBind	naváže DataSource na graf, který se bude generovat
RenderContents	spustí se při vykreslování ovládacího prvku
DetConstants	nastaví konstanty pro ovládací prvek

Třída Install

Třída install má na starosti vytvořit potřebné adresáře a soubory nejsou-li k dispozici

Metoda	Popis
Install	vše se vykoná v konstruktoru objektu

Třída AshMan

Třída Ashman má na starosti mazání starých grafů

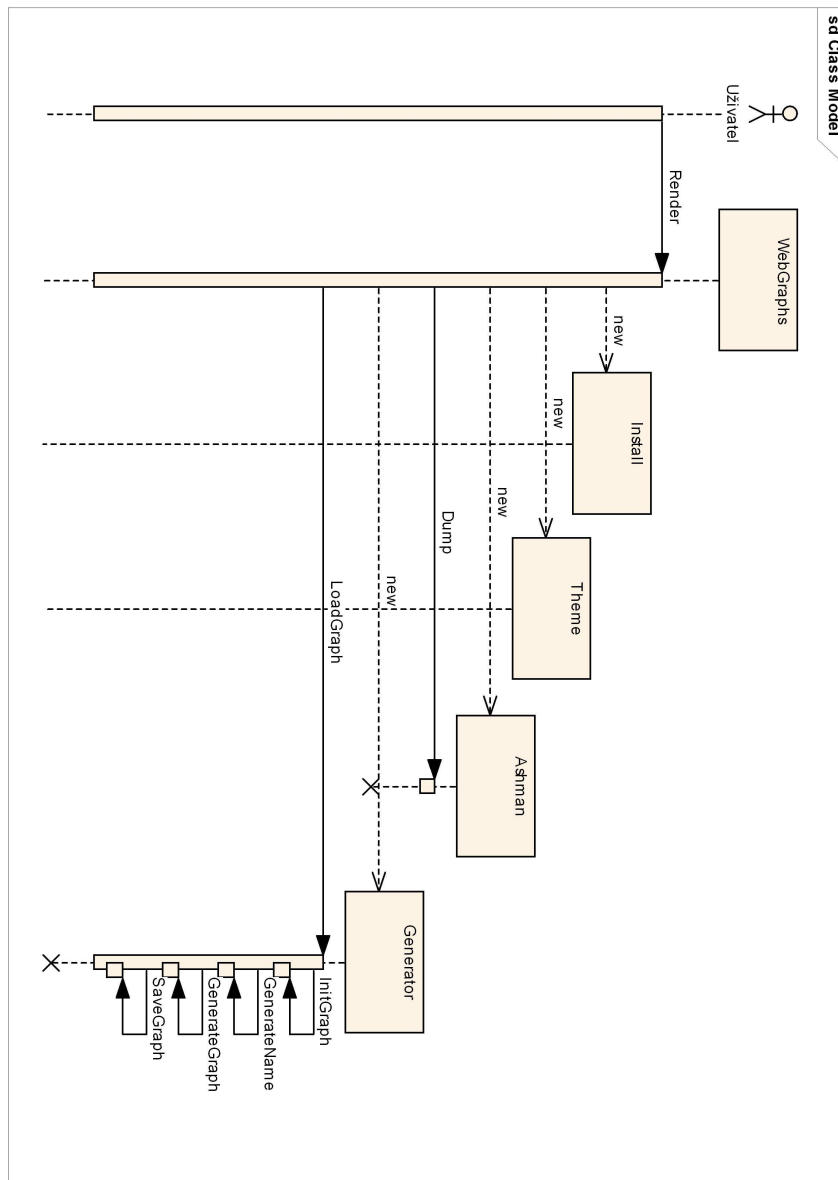
Metoda	Popis
Dump	smaže všechny staré grafy

Třída Generátor

Třída generátor má na starosti vygenerování grafu. Jsou v ní definovány metody, které spočítají body, vygenerují řetěz, který jednoznačně určí generovaný graf, vykreslí části grafu, graf uloží a vygenerují html kód pro zobrazení grafu na stránce.

Metoda	Popis
ConvertToPoint	ze seznamu objektů typu GPoint vytvoří pole objektů typu Point
CountPoint	spočítá funkční hodnoty v bodech a vrátí seznam bodů představující části grafu funkce, v ideálním případě vrací právě jeden seznam bodů. Bod představuje objekt typu GPoint
CountIEPoints	spočítá inflexní a extrémní body
DrawAxis	vykreslí osy grafu
DrawGraphFromPoint	vykreslí křivku podle zadaných bodů
DrawGraphs	vykreslí grafy ze seznamu grafů. Graf je objekt typu Graph
GenerateGraph	vygeneruje graf
GenerateName	vygeneruje řetěz, který jednoznačně určí generovaný graf
InitGraph	vytvoří obrázek, do kterého se bude zakreslovat generovaný graf
LoadGraph	vygeneruje kód pro vykreslení grafu na stránce
Resize	zvětší či zmenší generovaný graf
SaveExample	uloží graf do daného umístění
SaveGraph	uloží graf do adresáře temp
ShowPoint	vykreslí bod do grafu

4.3. Realizace



Obrázek 1. Diagram realizace generování grafu

5. Použití WebGraphs ve Visual Studiu 2005

5.1. Přidání do projektu

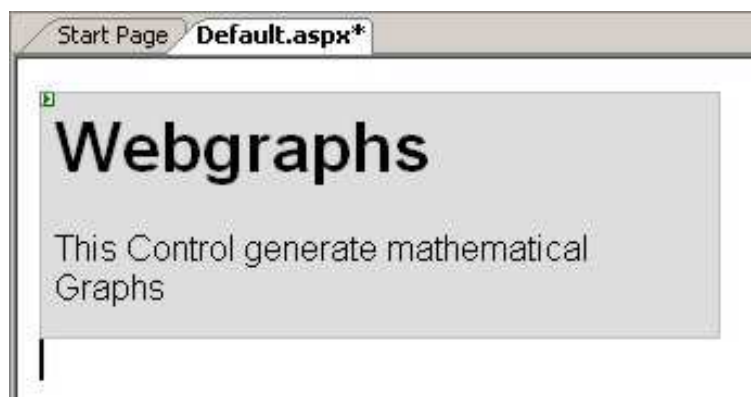
Nejdříve je nutné přidat serverový ovládací prvek do projektu, kde se bude používat. To lze udělat tak, že přidáme knihovny WebGraphs.dll a Eval3.dll do referencí a ovládací prvek zaregistrujeme na každou stránku, kde jej budeme chtít použít a to například takto:

```
<%@ Register Assembly="WebGraphs" Namespace="WebGraphs" TagPrefix="cc1" %>
```

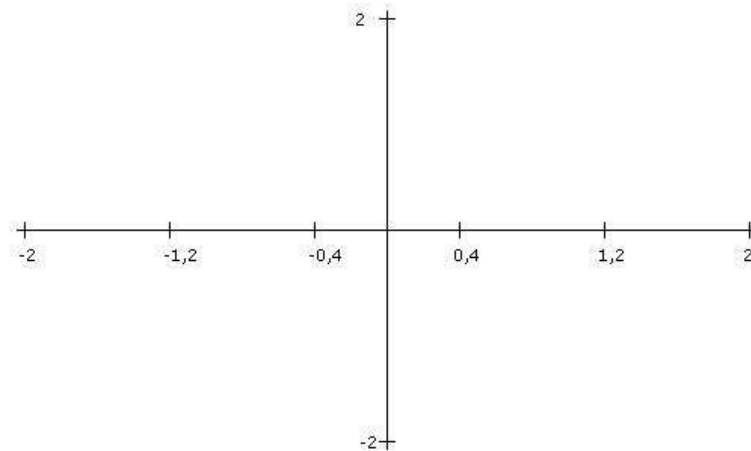
Bude-li si chtít programátor ulehčit práci, pak si může serverový ovládací prvek přidat rovnou do toolboxu a to tak, že klikne pravým tlačítkem na skupinu, kam chce ovládací prvek přidat a vybere z kontextové nabídky položku "Choose Items" a přidá tam knihovnu WebGraphs.dll.

5.2. Možnosti použití

Po vložení serverového ovládacího prvku na webový formulář je možno stránku ihned spustit. Objeví se prázdný graf a pokud nejsou u projektu vytvořeny potřebné adresáře a soubory, potom si je ovládací prvek sám vytvoří.

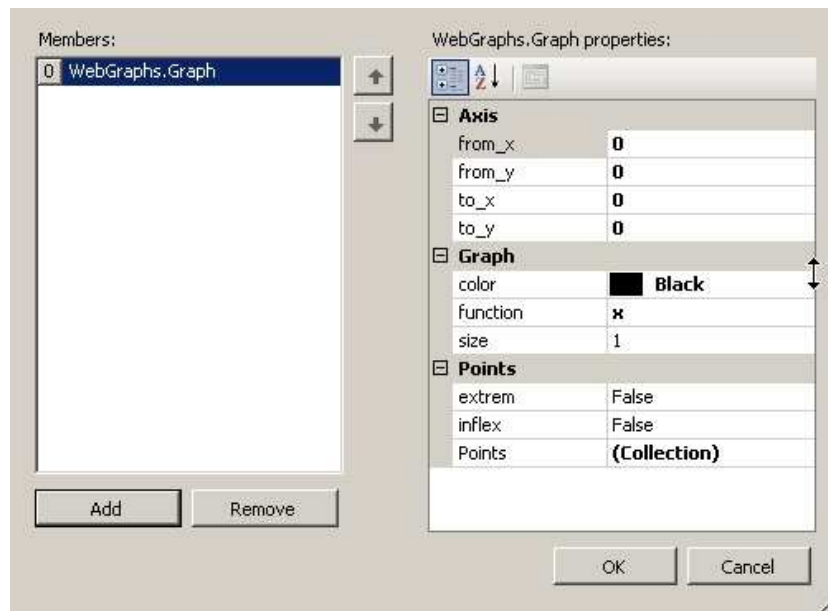


Obrázek 2. Serverový ovládací prvek webgraphs v návrhovém zobrazení)



Obrázek 3. Prázdný graf

Pokud chce programátor vygenerovat nějaký graf, pak stačí tento graf zadat. V návrhovém zobrazení postačí kliknout na atribut Graphs, kde vyskočí editor pro kolekci, ve kterém je možno přidat libovolné množství grafů.



Obrázek 4. Graph collection editor)

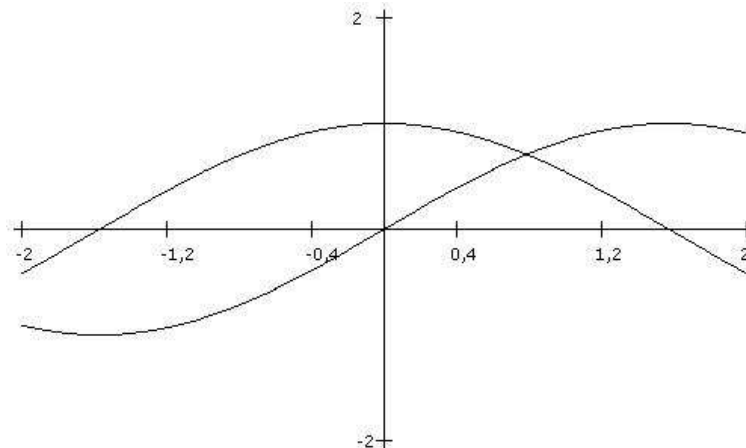
Jednoduchý příklad zdrojového kódu pro vykreslení grafů funkcí $\sin(x)$ a $\cos(x)$:

```
<cc1:WebGraphsControl ID="WebGraphsControl1" runat="server"
Width="500" Height="300">
```

```

<Graphs>
  <cc1:Graph color="Black" from_x="-2" from_y="-2"
    function="sin(x)" to_x="2" to_y="2">
  </cc1:Graph>
  <cc1:Graph color="Black" from_x="-2" from_y="-2"
    function="cos(x)" to_x="2" to_y="2">
  </cc1:Graph>
</Graphs>
</cc1:WebGraphsControl>

```



Obrázek 5. Graf funkci $\sin(x)$ a $\cos(x)$

Samozřejmě, že je možné vyplnit i všechny další atributy, které jsou popsány výše.

Další možností, jak zobrazit nějaký graf, je pomocí datové vazby. Stačí si vytvořit seznam bodů, ten navázat na `DataSource` a následně zavolat funkci `DataBind`. Příkladem může být následující zdrojový kód:

```

List<WebGraphs.GPoint> DataSource = new List<WebGraphs.GPoint>();

DataSource.Add = new WebGraphs.GPoint(0, 0);

...

DataSource.Add = new WebGraphs.GPoint(x, y);

WebGraphsControl1.DataSource = DataSource;
WebGraphsControl1.DataBind();

```

6. Webová aplikace demonstrující použití Web-Graphs

Pro demonstraci použití serverového ovládacího prvku webgraphs jsem vytvořil jednoduchou ASP.NET aplikaci. Jde pouze o jeden formulář, kam jsem přidal požadovaný ovládací prvek WebGraphsControl s nastavenou výškou a šířkou na 500 x 300 pixelů. Pro zadání grafu stačí dvě textové pole. Do prvního pole uživatel zadá funkci, kterou chce vygenerovat a do druhého zadá, odkud kam se má graf generovat. Potom může daný graf vygenerovat nebo ho přidat k dalším grafům. Podporu datových vazeb program demonstruje tak, že si uživatel může přidat alespoň dva body a dokud nestiskne tlačítko DataBind, na které je navázána metoda DataBind z ovládacího prvku, tak se nic neděje. Jakmile se na toto tlačítko zmáčkne, potom se zobrazí lomená čára, která prochází zadanými body.

Další funkčností je použití témat, která jsou k dipozici, proto aplikace vytáhne všechna uložená témata a zobrazí jejich náhledy. Na tyto náhledy může uživatel kliknout a tak změnit téma zobrazení grafu. Vše je uděláno tak, že se na místo, kde se mají zobrazit ony náhledy, vloží ovládací prvek Placeholder, do kterého při načítání stránky vložíme náhledy uložených témat jako ovládací prvky ImageButton a na událost onClick spustíme metodu, která nastaví vybrané téma.

```
DirectoryInfo dir = new
    DirectoryInfo(AppDomain.CurrentDomain.BaseDirectory + "/themes/");

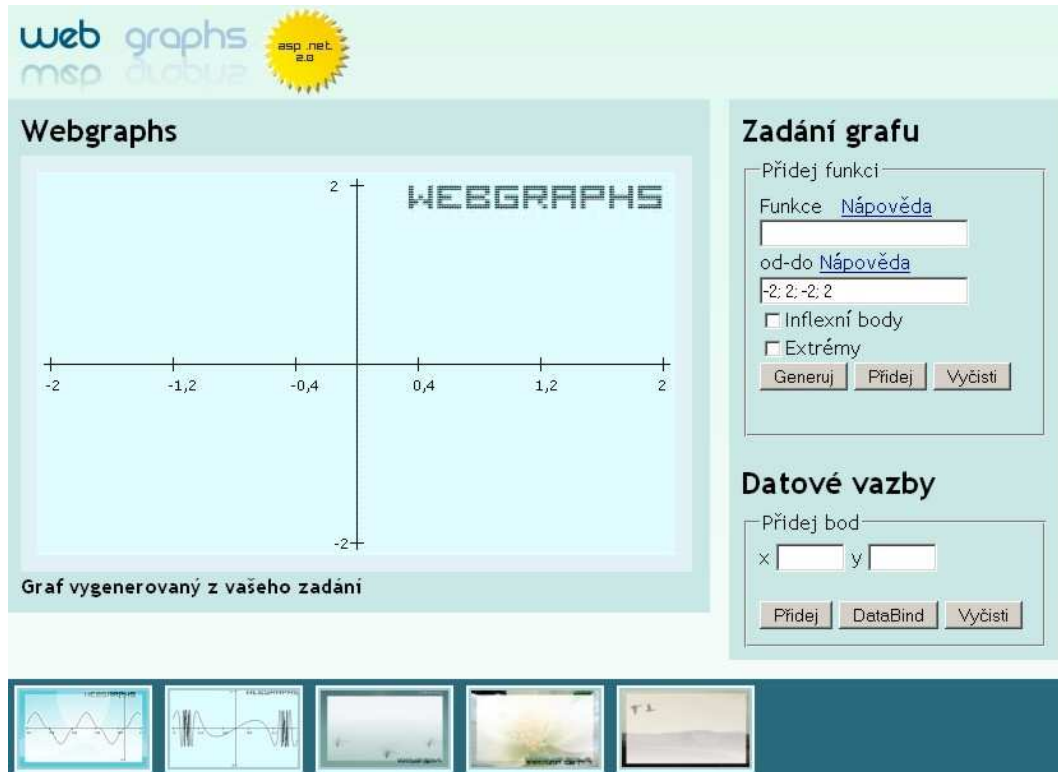
ImageButton obr;
string file;
XmlReader reader;

foreach (DirectoryInfo f in dir.GetDirectories())
{
    String name = f.Name;
    obr = new ImageButton();
    reader = XmlReader.Create(AppDomain.CurrentDomain.BaseDirectory
        + "/Themes/" + f.Name + "/theme.xml");
    reader.ReadToFollowing("theme");
    reader.ReadToFollowing("thumbnail");
    file = reader.GetAttribute("file");
    obr.ImageUrl = "themes/"+f.Name+"/"+file;
    obr.BorderStyle = BorderStyle.Solid;
    obr.BorderColor = System.Drawing.Color.White;
    obr.BorderWidth = Unit.Pixel(2);
```

```

obr.ID = f.Name;
obr.AlternateText = f.Name;
obr.Click += new ImageClickEventHandler(ObrClick);
Placeholder1.Controls.Add(obr);
}

```



Obrázek 6. Webová aplikace demonstrující použití Serverového ovládacího prvku

Tato aplikace je k dispozici na webu na adrese <http://horak.asp2.cz>, kde si ji můžete vyzkoušet.

Závěr

V této diplomové práci jsem analyzoval problematiku webového programování, popsal principy a mechanismy tvorby serverových ovládacích prvků v ASP.NET 2.0 a naprogramoval jeden takový serverový ovládací prvek generující matematické grafy v 2D. Vytvořený serverový ovládací prvek by mohl být užitečným pomocníkem při zobrazování grafů matematických funkcí na webu, bude volně k dispozici včetně zdrojových kódů k libolnému použití.

Reference

- [1] Matthew MacDonald, Mario Szpuszta, *ASP.NET 2.0 a C#*
- [2] Andrew Troelsen, *C# a .NET 2.0*
- [3] Microsoft MSDN
<http://msdn.microsoft.com/>
- [4] Wikipedia
<http://www.wikipedia.org/>