

Programovací jazyk C# Další kapitoly

Ing. Marek Běhálek
Katedra informatiky FEI VŠB-TUO
A-1018 / 597 324 251
<http://www.cs.vsb.cz/behalek>
marek.behalek@vsb.cz

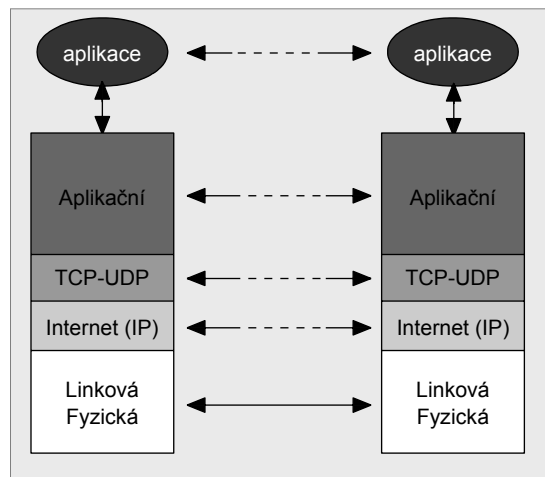


Náplň kapitoly

- V této kapitole budete seznámeni se základy programování v těchto oblastech.
 - Práce v síti
 - Práce s XML
 - Windows Forms
 - Webové aplikace využívající ASP.NET



Úvod do sítí - TCP/IP model - realita



Další kapitoly

3

Úvod do sítí - Komunikace dvou aplikací



- IP adresa – identifikace počítače
- port – identifikace aplikace v počítači



Další kapitoly

4

Úvod do sítí - IP adresa



- Internet Protocol address
 - jednoznačně identifikuje zařízení v IP síti
 - IPv4
 - 32 bitů - (4 bajty v desítkové soustavě)
 - *aaa.bbb.ccc.ddd*
 - 158.196.149.9
 - IPv6
 - 128 bitů - 16 bajtů (8 dvojbajtů v šestnáctkové soustavě)
 - *aaaa:bbbb:cccc:dddd:eeee:ffff*
 - 2001:0db8:85a3:08d3:1319:8a2e:0370:7334
- DNS (Domain Name System)
 - *www.vsb.cz => 158.196.149.74*
 - *dom_k...dom₃.dom₂.dom₁*

Další kapitoly

5

Úvod do sítí - World Wide Web



- množství hypertextových dokumentů
- vyhledávání, prohlížení, sdílení dokumentů
- hypertext
 - odkaz v dokumentu na kterýkoliv dokument nebo část dokumentu kdekoliv ve WWW
- HTTP (HyperText Transfer Protocol)
 - protokol pro přenos dat mezi klientem a webovým serverem
- HTML (HyperText Markup Language)
 - systém textových značek pro prohlížeč
 - značky definují odkazy, způsob formátování dat a pod.

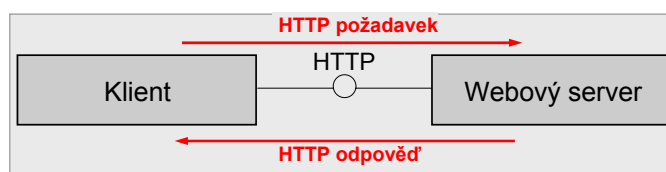
Další kapitoly

6

Úvod do sítí - World Wide Web



- Webový server
 - aplikace poskytující dokumenty
 - HTTP (HyperText Transfer Protocol)
 - HTTP 1.0 bezstavový protokol
 - HTTP požadavek (GET, POST, HEAD)
 - HTTP odpověď



7

Úvod do sítí - HTTP 1.0



- bezstavový protokol
 - požadavek – odpověď => ukončení spojení
 - požadavek i odpověď mají následující formát

```
<úvodní řádek>
<hlavička-1>: <hodnota-1>
<hlavička-2>: <hodnota-2>
...
<hlavička-n>: <hodnota-n>
<prázdný řádek>
<nepovinné tělo zprávy>
```

Další kapitoly

8

Úvod do sítí - HTTP požadavek



- Požadavek GET
 - GET *cesta* HTTP/*verze*

```
GET / HTTP/1.0
Host: www.google.com
User-Agent: Mozilla/5.0
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: cs-CZ,cs;q=0.9,en-US;q=0.8,en;q=0.7,defaultQLS
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1250,utf-8;q=0.7,*;q=0.7
Cookie:
PREF-ID=c0f4d58d41001453:TB=2:TM=1168255510:LM=1177510598:S=32VaTkcUR4ijOcQr
```

- Požadavek POST
 - POST *cesta* HTTP/*verze*

```
POST /path/script.cgi HTTP/1.0
From: mole@garden.cs
User-Agent: MoleHill/0.13
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

name=mole&event=trap&action=kill
```

Úvod do sítí - HTTP odpověď



- HTTP/*verze* *kód text*

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: gws
Content-Length: 68
Date: Fri, 21 Sep 2007 08:53:37 GMT
.....W.v.6...S.Z.jI..8.J".6q..I.'Mw.??>.R.@...G...../.....
```

```
HTTP/1.0 404 Not Found
...
...
```

kód	význam
1xx	informační zpráva
2xx	indikuje nějaký úspěch
3xx	redirekce klienta na jinou URL
4xx	chyba na straně klienta
5xx	chyba na straně serveru

Další kapitoly

10

Práce v síti - Jazyk C# a podpora sítí



- Podpora práce v síti je implementována v knihovnách prostředí .NET
 - System.Net
 - System.Net.Sockets
- Jazyk C# (.NET Framework Class Library) implementuje podporu standardních protokolů jako:
 - TCP
 - UDP
 - HTTP
 - HTTPS
 - file

Další kapitoly

11

Práce v síti - Obecná architektura: požadavek/odpověď



- Používá URI (Uniform Resource Indicator).
- Přenos dat je realizován I/O proudy.
- Požadavek je realizován pomocí abstraktního bázevého typu *WebRequest*.
 - Konfigurace požadavku.
 - Převzetí výsledku.
- Odpověď je realizována rovněž pomocí abstraktní třídy *WebResponse*.
 - Využívá I/O proud (NetworkStream).

Další kapitoly

12

Práce v síti - Podpora HTTP



- Popsaná architektura požadavek/odpověď vnitřně podporuje rozšíření pro jednotlivé protokoly.
- Stačí vhodně "převést" získanou odpověď.

```
class HttpTest
{
    static void Main(string[] args)
    {
        WebRequest request =
        WebRequest.Create("http://www.cs.vsb.cz");
        request.Method="HEAD"; //only heads;
        HttpWebRequest httpRequest = (HttpWebRequest)request;
        httpRequest.UserAgent = "CSApplication/1.0";

        WebResponse response = request.GetResponse();
        HttpWebResponse httpResponse =
        (HttpWebResponse) response;
        Console.WriteLine(httpResponse.Server);
    }
}
```

Další kapitoly

13

Práce v síti - WebClient



- Poskytuje rozhraní k síťovým prvkům na vyšší úrovni než WebRequest.
 - Získání dat:
 - DownloadData()
 - DownloadFile()
 - Odesílání dat:
 - UploadData()
 - UploadFile()

```
class Example
{
    static void Main(string[] args)
    {
        WebClient client = new WebClient();
        byte[] buffer =
        client.DownloadData("http://www.cs.vsb.cz");
        string document = Encoding.ASCII.GetString(buffer);
        Console.WriteLine(document);
    }
}
```

Další kapitoly

14

Práce v síti - Podpora protokolů TCP a UDP



- namespace System.Net.Sockets
- Základním typem je třída Socket.
 - TCP
 - TcpListener - naslouchá na příchozím spojení a vytváří instance Socket.
 - TcpClient - základní I/O operace přes síť.
 - UDP
 - UdpClient - zajišťuje podporu jak vysílání tak přijímání UDP datagramů.

Další kapitoly

15

Práce v síti - Podpora protokolů TCP a UDP



```
class TCPTest
{
    static string message="Hello world!"
    static void Main(string[] args)
    {
        TcpListener l = new TcpListener();
        l.Start();
        while(true) {
            Socket s = l.AcceptSocket();
            byte[] barr =
Encoding.ASCII.GetBytes(message.ToCharArray());
            s.Send(barr);
            s.Shutdown(SocketShutdown.Both);
            s.Close();
        }
    }
}
```

Další kapitoly

16

Práce v síti - DNS



- Existují třídy pro překlad IP adres na DNS záznamy a naopak.

```
using System;
using System.Net;

class DnsExample
{
    static void Main(string[] args)
    {
        IPEndPoint he = Dns.GetHostByName("www.google.com");
        IPAddress[] addressList = he.AddressList;
        foreach(IPAddress address in addressList)
            Console.WriteLine(address);
    }
}
```

Další kapitoly

17

Základy XML - úvod



- eXtensible Markup Language
- množina pravidel
 - sémantické značky (tagy, elementy)
 - rozdělují dokument na části
 - identifikuje části dokumentu
- jazyk pro popis jazyků
 - meta-značkový jazyk
 - definuje syntaxi definice jiného jazyka
- vychází se SGML (Standard Generalized Markup Language)
 - stejné možnosti
 - jednoduchost

Další kapitoly

18

Základy XML - další značkovací jazyk?



- xml není jazyk, je to o **meta-jazyk**
- značky
 - vytvářeny podle potřeby
 - jména podle významu
 - například:
 - jména tabulek databáze
 - jména atributů
 - ...

Další kapitoly

19

Základy XML - značení



- značení odlišuje XML od čistého textu
- většina značení jsou tagy (značky)
 - tag je vše co začíná znakem '<' a končí znakem '>'
 - tag má jméno
 - musí začínat [a-z, A-Z, _]
 - je case sensitive (a jsou různé)
- prázdný tag
 - nemá obsah
 - možnost použití zkratky pomocí koncovky '/>'
<empty />

Další kapitoly

20

Základy XML - Pravidla tvorby XML dokumentu



- dokument musí začít XML deklarací
- element s daty musí mít startovní a koncový tag
- elementy s jediným tagem končí “/”
- musí existovat jediný kořenový element
- elementy se nesmí překrývat
- atributy v uvozovkách
- “<” a “&” jen u tagů a znakových entit
- ...

Další kapitoly

21

Základy XML - Atributy



- počáteční a prázdné tagy
- dvojice jméno = hodnota
- jméno
 - musí začínat [a-z, A-Z, _]
 - stejné jméno v tagu jen jednou
- hodnota
 - řetězec v uvozovkách (nebo apostrofech)
 - libovolné znaky

```
<task name="cdrecord" owner="gyp35" priority="-19" />
<par indent='0.5' />
<par indent="0.5";" />
```

22

Základy XML - Komentáře



- začínají “<!--” a končí “-->”
- komentáře jsou ignorovány
- komentáře nemají
 - se nalézat před XML deklarací
 - být uvnitř tagu
 - obsahovat dvojici pomlček “--”
- komentáře by měly být používány k ohraničení a skrytí tagů

```
<!-- priklad komentare -->
```

23

Základy XML - Vkládání čistého textu



- sekce **CDATA**
 - obsahuje text bez interpretace značení
 - nesmí obsahovat dvojici “]]”

```
<![CDATA[  
for (int i = 0; i < array.length && error == null; i++)  
]]>
```

```
for (int i = 0; i &lt;t; array.length &amp;&amp; error == null; i++)
```

Další kapitoly

24

Základy XML - Document Type Definition (DTD)



- určuje XML dokumentu
 - seznam elementů, atributů, notací a entit
 - vztahy mezi nimi
 - strukturu
- nalézá se
 - prologu za deklarací
 - před prvním elementem
- buď přímo DTD nebo URL s DTD
- každý tag XML musí být deklarován v DTD
 - deklarace určuje jméno a obsah elementu

Další kapitoly

25

Práce s XML - Přístup k XML



- Použití abstraktní bázové třídy s konkrétní implementací úložišť.
- Podobný mechanismus jako u I/O vstupů a výstupů.
- Abstraktní bázové třídy jsou:
 - XmlReader;
 - XmlWriter.
- Typy pocházejí z oboru názvů:
 - System.Xml
 - System.Xml.XPath

Další kapitoly

26

Práce s XML - XmlReader



- Poskytuje schopnost číst XML dokumenty.
- Abstraktní třída, z níž při práci s určitým XML zdrojem je třeba vytvořit podtřidu.
- Existují 3 konkrétní implementace XmlReaderu:
 - XmlTextReader
 - analyzuje XML z libovolného proudu textu.
 - XmlNodeReader
 - analyzuje XML z XmlNode.
 - XmlValidatingReader
 - XmlReader vykonávající DTD nebo ověřující schéma analyzovaného dokumentu.

Další kapitoly

27

Práce s XML - XmlReader



- Většinou pro práci s XML stačí pouze XmlTextReader
- ```
//XmlTextReader can open an XML file
XmlTextReader tr = new XmlTextReader("xmlleg.xml");
```
- Předchozí příklad s použitím obecnějšího konstruktoru.
    - Lze číst XML kód z libovolného zdroje typu Stream včetně URL adres a databázových dat.

```
//XmlTextReader uses XmlReader for readin data from
// thestream.
XmlTextReader tr = new XmlTextReader(
 new TextReader(
 new FileStream("xmlleg.xml", FileMode.Open)));
```

Další kapitoly

28

## Práce s XML - XmlReader



- Schopnost číst data z proudu umožňuje zpracovávat XML data bez nutnosti je nejprve uložit do souboru.
- XmlReader slouží jako kurzor ukazující na různé konstrukce XML obsažené v proudu jako elementy, atributy, atd.
- K aktuální konstrukci lze přistupovat vlastnostmi:
  - Name - vrací kvalifikovaný název elementu nebo atributu;
  - Value - vrací hodnotu prostého textu.
- K navigaci se používají metody:
  - Read - k iterování a přístupu k dalšímu elementu proudu;
  - MoveXXX - k navigaci v proudu (například MoveToNextElement)
  - Navigace je jen dopředná, mineme-li uzel, nelze se k němu vrátit.

Další kapitoly

29

## Práce s XML - Tvorba XML dokumentu



- Lze vytvořit XML dokument přímo přidáním XML tagů:
  - `string xml = "<greeting>" + greetingOfTheDay + "</greeting>"`
- Tato technika přináší řadu problémů jako překlepy a náhodné chyby.
  - špatné zformování dokumentu
  - znemožnění analýzy XML dokumentu
- Třída XmlWriter umožňuje XML dokument generovat.

Další kapitoly

30

## Práce s XML - XmlWriter



- Abstraktní typ pro vytváření dat odpovídající specifikaci XML.
  - Má implementace podobně jako XmlReader.
  - Jako XmlReader má XmlTextReader, tak XmlWriter má XmlTextWriter.
- Některé metody a vlastnosti XmlTextWriteru:
  - Formatting - určuje zda se mají například odsazovat úrovně;
  - Indentation - určuje kolik znaků se má odsazovat každá úroveň;
  - WriteStartElement - vytvoří počáteční tag zadaného elementu;
  - WriteEndElement - vytvoří koncový tag odpovídajícího elementu;
  - Write - zapíše do dokumentu prostý text.

## Práce s XML - XmlWriter



- Příklad vygenerování XML dokumentu z předchozího příkladu včetně deklarace dokumentu:

```
XmlTextWriter xw = new XmlTextWriter("greetings.xml");
xw.Formatting = Indented;
xw.Indentation = 2;
xw.WriteStartDocument();
xw.WriteStartElement("greeting");
xw.Write(greetingOfTheDay);
xw.WriteEndElement();
xw.WriteEndDocumen
```



## Práce s XML – XML Document



- Čtení a zápis XML dat uvedený výše je dosti omezující.
- Například u XmlReaderu pohyb XML dokumentem pouze vpřed.
- .NET obsahuje třídu XmlDocument k modelování celého dokumentu XML.

```
XmlDocument doc = new XmlDocument();
doc.Load(new XmlReader(...))
```

Další kapitoly

33

## Práce s XML – XML Document



- Jakmile je XmlDocument naplněn daty lze z něj získat informace.
- Vlastnost DocumentElement vrátí element dokumentu:

```
XmlNode docNode = doc.DocumentElement();
//printing all first level child nodes of the docNode node
foreach(XmlNode n in docNode.ChildNodes)
 Console.WriteLine(n.Name);
```

- Od tohoto okamžiku se navigace stává pouhým používáním vlastností a metod XmlNode.
- Vlastnost ChildNodes vrací instanci XmlNodeList obsahující dceřiné uzly daného uzlu.
- Vlastnosti Name a Value vracejí různé informace v závislosti na typu aktuálního uzlu.

Další kapitoly

34

## Práce s XML – XML Document



- Některé vlastnosti třídy XMLNode
  - Value
  - Name
  - InnerText
  - ParentNode
  - HasChildNodes, ChildNodes
  - Attributes
- Některé metody třídy XMLNode
  - GetEnumerator
  - AppendChild, RemoveChild, ReplaceChild, RemoveAll,...
  - Normalize
  - WriteTo, WriteContentTo
- Pro procházení lze také využít indexer

Další kapitoly

35

## XPath



- jazyk pro adresování částí XML dokumentu
- základní operace pro práci s řetězci, čísla, ...
- Cesta (Location Path)
  - sekvence kroků oddělených lomítkem „/“
  - každý krok určuje množinu uzlů relativních ke *kontextovému uzlu* s použitím:
    - os (axis)
    - testů uzlů (node test)
    - predikátů

Další kapitoly

36



# XPath - cesta

- analogie s cestou v souborovém systému

/devcata/pritelkyne/hezke/chytre/patricie.jpg

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
 <cd country="USA">
 <title>Empire Burlesque</title>
 <artist>Bob Dylan</artist>
 <price>10.90</price>
 </cd>
 <cd country="UK">
 <title>Hide your heart</title>
 <artist>Bonnie Tyler</artist>
 <price>9.90</price>
 </cd>
 <cd country="USA">
 <title>Greatest Hits</title>
 <artist>Dolly Parton</artist>
 <price>9.90</price>
 </cd>
</catalog>
```

- /catalog
- /catalog/cd
- /catalog/cd/price
- //cd
- /catalog/cd/\*
- /catalog/\*/price
- /\*\*/price

/step/step/...

step/step/...



# XPath - cesta

/catalog/cd[1]

/catalog/cd[last()]

/catalog/cd[price]

/catalog/cd[price>10.80]

/catalog/cd[price=10.90]/price

//title | //artist | //price

/step/step/...

step/step/...

//@country

//cd[@country]

//cd[@\*]

//cd[@country='UK']

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
 <cd>
 <title>Empire Burlesque</title>
 <artist>Bob Dylan</artist>
 <price>10.90</price>
 </cd>
 <cd country="UK">
 <title>Hide your heart</title>
 <artist>Bonnie Tyler</artist>
 <price>9.90</price>
 </cd>
 <cd country="USA">
 <title>Greatest Hits</title>
 <artist>Dolly Parton</artist>
 <price>9.90</price>
 </cd>
</catalog>
```

Další kapit

## Práce s XPath – Jednoduchý příklad



- XmlNode implementuje metodu SelectNodes

```
XmlTextReader tr = new XmlTextReader(new
 StringReader(xmlContent));
XmlDocument doc = new XmlDocument();
doc.Load(tr);
XmlNode docElement = doc.DocumentElement;

XmlNodeList result =
 docElement.SelectNodes("/book/authors/author/text()");
foreach(XmlNode n in result){Console.WriteLine(n.Value);}
```

Další kapitoly

39

## Práce s XPath - XPathNavigator



- Umožňuje položit dotaz formou výrazu XPath.
- Rozhraní IXPathNavigable.
  - Toto rozhraní definuje jedinou metodu CreateNavigator.
  - Všechny objekty které implementují rozhraní mohou vrátit instanci XPathNavigator.
  - XmlNode, XPathDocument
- Umožňuje spustit dotaz a vrátit instanci třídy XPathNodeIterator.
- Na každém dokumentu lze vytvořit několik navigátorů a nezávisle jimi pohybovat.
- Při realizaci algoritmů lze například pozici navigátorů porovnávat a zjistit, zda například ukazují na stejnou pozici.

Další kapitoly

40

## Práce s XPath - XPathNavigator



- **Jednoduchý příklad použití:**

```
XPathDocument doc = new XPathDocument("books.xml");
XPathNavigator nv = doc.CreateNavigator();

XPathNodeIterator iter = nv.Select("/book/authors/author");

while(iter.MoveNext()){
 Console.WriteLine("Autor:" + iter.Current.Value);
}
```

- **XPathNodeIterator**

- **Iterátor schopen procházet kolekcí uzlů vyhovujících specifikované podmínce.**

Další kapitoly

41

## Práce s XPath - XPathNavigator



- **Některé vlastnosti třídy XPathNavigator**

- IsEmpty
- IsNode
- Name
- XmlType
- HasAttributes
- HasChildren

- **Některé metody třídy XPathNavigator**

- Evaluate
- CreateAttribute, MoveToNextAttribute
- AppendChild

Další kapitoly

42



## XSL

- eXtensible Stylesheet Language
- XML nemá předdefinované značky
  - HTML  
`<table>...</table>`
  - XML  
`??? <table>...</table> ???`
- říká, jak má být XML dokument zobrazen
- části:
  - XSLT - jazyk pro transformaci XML dokumentu
  - XPath - jazyk pro určení částí XML dokumentu

Další kapitoly

43



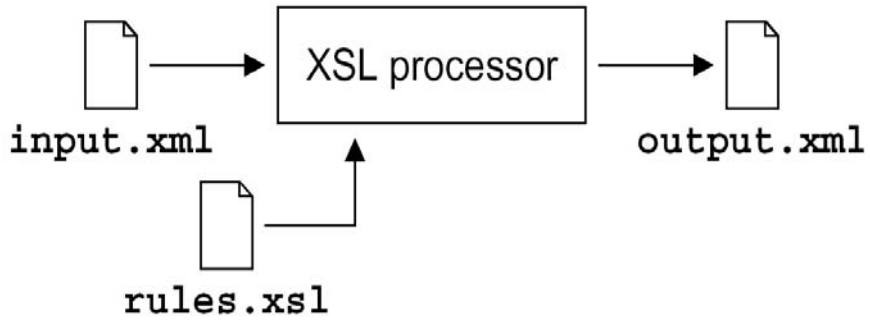
## XSLT

- nejdůležitější část standardu XSL
- transformuje XML do XML (XHTML, HTML)
- vkládá/filtruje elementy do výstupního dokumentu
- třídí a uspořádává elementy
- na základě testu lze rozhodovat, co udělat s danými elementy
- transformuje zdrojový XML strom na cílový XML strom
- používá XPath k nalezení odpovídajících vzorů k transformaci

Další kapitoly

44

# XSLT



Další kapitoly

45

# XSLT – příklad 1



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th align="left">Title</th>
<th align="left">Artist</th>
</tr>
<tr>
<td>
<xsl:value-of select="catalog/cd/title"/>
</td>
<td>
<xsl:value-of select="catalog/cd/artist"/>
</td>
</tr>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```



Další kapitoly

## XSLT – příklad 2

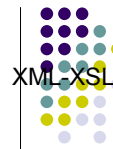


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

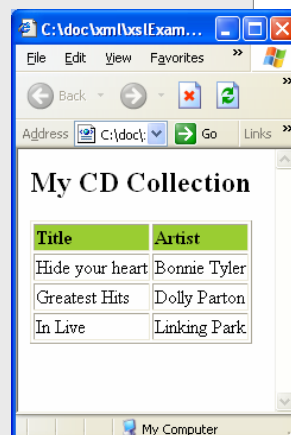
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th align="left">Title</th>
<th align="left">Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<tr>
<td>
<xsl:value-of select="title"/>
</td>
<td>
<xsl:value-of select="artist"/>
</td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```



## XSLT – příklad 3



```
<xsl:template match="/">
...
...
<xsl:for-each select="catalog/cd">
<xsl:sort select="artist"/>
<xsl:if test="price < 10">
<tr>
<td>
<xsl:value-of select="title"/>
</td>
<td>
<xsl:value-of select="artist"/>
</td>
</tr>
</xsl:if>
</xsl:for-each>
...
</xsl:template>
</xsl:stylesheet>
```





## Práce s XSTL - Transformování dokumentů



- Příklad použití XML transformace v jazyce C#

```
//obtaining pattern for transformation
XsltTransform transform = new XsltTransform();
transform.Load("pattern.xml");

//obtaining data for transformation
XmlTextReader tr = new XmlTextReader("source.xml");

//loading XML document to XmlDocument
XmlDocument doc = new XmlDocument();
doc.Load(tr);

//defining where to put output
StringWriter output = new StringWriter();

//transformation
transform.Transform(doc, null, output);
```

Další kapitoly

49

## WinForms



- Jmenné prostory
  - System.Windows.Forms - Formuláře, dialogy, okna,...
  - System.Drawing - Kreslení, GDI+
- Základem je třída `Application`
  - `Application.Run(Form)` - Zobrazí předaný formulář a běží dokud nedojde k uzavření formuláře.
  - Ukončení běhu aplikace
    - `Application.Exit()`
    - `Application.ExitThread()`
  - Třída obsahuje celou řadu dalších událostí (`ApplicationExit`, `Idle`) a metod (`ExecutablePath`, `StartupPath`).

Další kapitoly

50

# WinForms – Návrh Windows Forms



- **Forms**
  - Form – reprezentuje jakékoliv okno v aplikaci
  - Vlastnost **BorderStyle** definuje, jaké okno se objeví:
    - Standard, Tool, Borderless, Floating Window
  - Form může obsahovat další formuláře (Form) = MDI (Multiple Document Interface)
  - Modální formulář
    - Nedovolí uživateli manipulovat s původním oknem dokud není nové okno uzavřeno
- **Controls**
  - Standardní ovládací prvky jako Button, Label, Radiobutton, TextBox, ...
  - Specifické ovládací prvky jako DataGridView, MonthCalendar, ...
  - Uživatelsky definované ovládací prvky.

Další kapitoly

51

# WinForms – HelloWorld



```
class HelloWorldForm : Form {
 Label lab;

 HelloWorldForm() {
 this.Text = "HelloWorldForm Titel";
 this.Size = new Size(200,100);
 lab = new Label();
 lab.Text = "HelloWorld";
 lab.Location = new Point(20, 20);
 this.Controls.Add(lab);
 }

 public static void Main(string[] argv) {
 Application.Run(new HelloWorldForm());
 }
}
```

Další kapitoly

52

## WinForms - Form

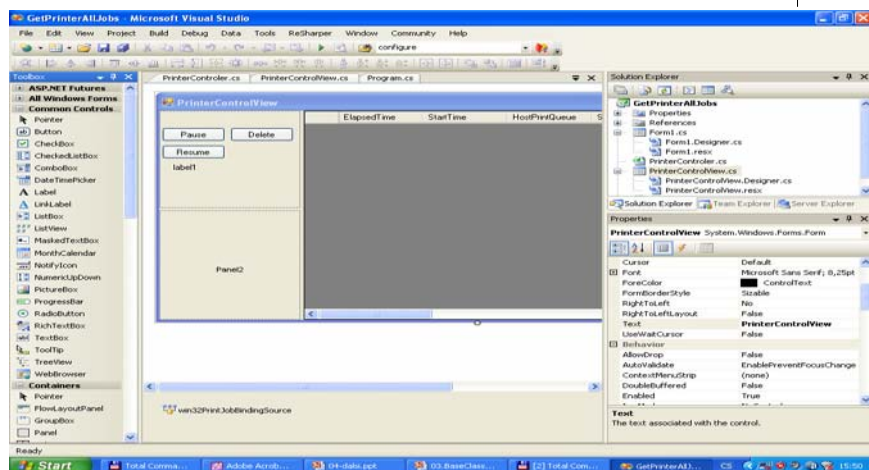


- Některé události třídy Form
  - Load, Activated, Deactivate, Closing, Closed
- Některé vlastnosti třídy Form
  - Location, Size, MinimumSize, MaximumSize, Text, BackColor, ForeColor, FormBorderStyle, MinimizeBox, MaximizeBox, HelpButton, Icon, ShowInTaskBar, Opacity, Region, TopMost, WindowState
- Designer
  - Je možno použít pro generované formuláře.
  - Metoda InitializeComponent
  - Ve VS2005 je vygenerovaný kód uložen do jiného souboru (xxx.Designer.cs).
- Formulář se při zavření pouze skryje – lze ho znova použít.

Další kapitoly

53

## WinForms - Designer



Další kapitoly

54

## WinForms - Control



- Třída Control je základní pro všechny ovládací prvky.
  - Poskytuje možnosti pro nastavení velikosti, pozice a stylu atd.
  - Vnořování ovládacích prvků,
    - Property **ControlCollection Controls**.
  - Zobrazení/skrytí, reakce na vstupy a zajištění vlákenní bezpečnosti.
- Některé vlastnosti třídy Control
  - Text, Visible, ForeColor, BackColor
- Některé události související se vzhledem
  - FontChanged, ForeColorChanged, Resize, Move

Další kapitoly

55

## WinForms – Reakce na vstupy



- Události
  - Klávesnice
    - KeyDown -> KeyPress -> KeyUp
  - Myši
    - Click a DoubleClick
    - MouseMove, MouseDown, MouseUp,
    - MouseEnter, MouseHover, MouseLeave
- Způsob zpracování události:
  - Událost získá prvek který má „focus“.

```
public delegate void EventHandler(object sender, EventArgs e);
```

```
Button b = new Button();
b.Click += new EventHandler(clickHandler);
```

```
private void clickHandler(object sender, EventArgs evArgs) { ... }
```

Další kapitoly

56

## WinForms – Návrh „Layoutů“



- Existují tři způsoby jak definovat Layout.
  - Anchor – Rozměry mezi ovládacími prvky zůstávají konstantní.
    - Úchyt - ke změně velikosti dochází pouze tehdy je-li ovládací prvek uchycen ze dvou stran
    - enum AnchorStyles - None, Left, Right, Top, Bottom
      - Možno kombinovat
  - Dock
    - Přilepí ovládací prvek na některou ze stran nadřazeného prvku.
    - enum DockStyle - None, Bottom, Left, Top, Right, Fill
  - Uživatelsky definované
  - V .NET 2.0 přibyl:
    - FlowLayoutPanel
    - TableLayoutPanel

Další kapitoly

57

## WinForms – Bezpečnost u vícevláknových aplikací



- Vlákno nesmí přistoupit k ovládacím prvkům vlastněným jiným vláknem.
  - Řešením této situace je požádat spuštění příslušné části kódu vláknem které ovládací prvek vlastní.
    - Lze využít jak synchronní tak asynchronní řešení.
  - Týká se jen nebezpečných akcí jako je zápis dat nebo čtení dat u nichž je vyžadována vzájemná konzistence.

```
if (this.InvokeRequired) {
 InitDelegate d=new InitDelegate(Reallnit);
 this.Invoke(d, server, args);
}
```

Další kapitoly

58

## GDI + (1)



- Podpora grafických operací v .NET Framework.
  - Jde o skupinu tříd zaobalující práci s GDI a poskytující jednotný přístup ze všech programovacích jazyků .NET.
  - Všechny třídy jsou dostupné ve jmenných prostorech System.Drawing a System.Drawing2D.
- 2D rozšíření:
  - Alfa-blending - nastavení průhlednosti objektu
  - Anti-aliasing - vyhlazení křivek
  - Gradientní a texturové vyplňování
  - Kardinální spline křivky - sekvence jednoduchých křivek spojených do jedné větší
  - Scalable-regions - škálovatelné regiony GDI
  - Transformace - rotace, posuvy, zvětšení, ...

Další kapitoly

59

## GDI + (2)

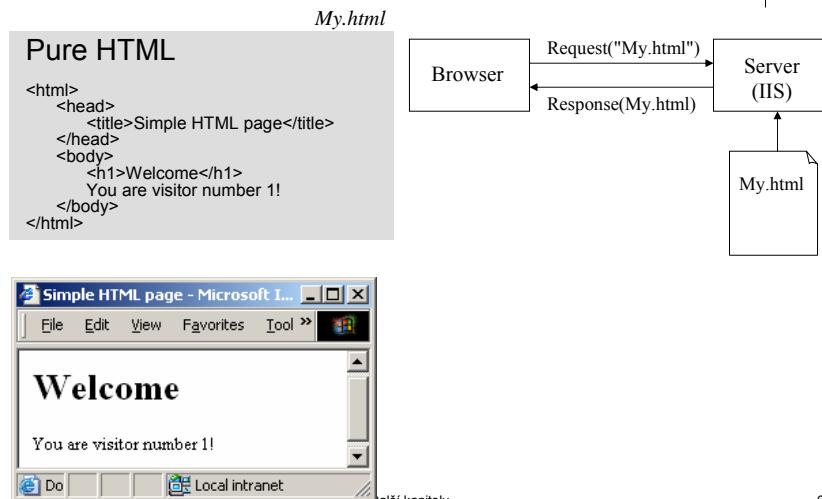


- Podpora práce s obrázky:
  - nativní formáty - podpora obrázků .jpg .gif .png .bmp .tiff .exif .icon
  - zpracování obrazu - změna jasu, kontrastu, vyvážení barev, rozmazání, ...
- Typografická podpora:
  - ClearType fonty a antialiasing
  - Textury a fonty - písmena textu mohou být texturovaná
  - Unicode - podpora práce s Unicode znaky a řetězci

Další kapitoly

60

# ASP.NET – Statická webová stránka



# ASP.NET – Dynamická ASPX stránka



*Counter.aspx*

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.IO" %>
<html>
<head> <title>Page counter</title> </head>
<body>
<h1>Welcome</h1>
You are visitor number <%
FileStream s = new FileStream("c:\\Data\\Counter.dat",
FileMode.OpenOrCreate);
int n;
try {
BinaryReader r = new BinaryReader(s);
n = r.ReadInt32();
} catch { n = 0; } // if the file is empty
n++;
s.Seek(0, SeekOrigin.Begin);
BinaryWriter w = new BinaryWriter(s);
w.Write(n); s.Close();
Response.Write(n);
%>
</body>
</html>
```

Dašší kapitoly

62

# ASP.NET – Code Behind



Counter.aspx

```
<%@ Page Language="C#" Inherits="CounterPage" CodeFile="CounterPage.cs" %>
<html>
 <head> <title>Page counter</title> </head>
 <body>
 <h1>Welcome</h1>
 You are visitor number <%=CounterValue()%> !
 </body>
</html>
```

CounterPage.cs

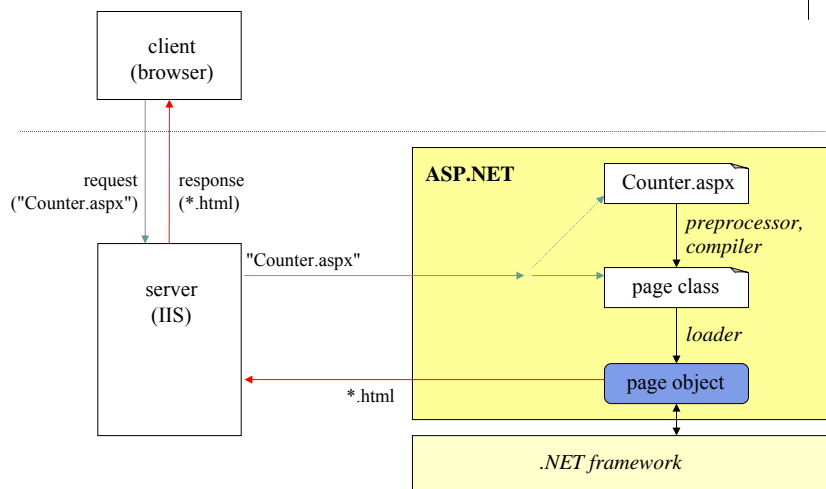
```
using System.IO;

public partial class CounterPage : System.Web.UI.Page {
 public int CounterValue() {
 FileStream s = new FileStream("c:\\Data\\Counter.dat", FileMode.OpenOrCreate);
 ...
 n = r.ReadInt32();
 n++;
 ...
 return n;
 }
}
```

Další kapitoly

63

# ASP.NET – Co se děje na pozadí?



Další kapitoly

64



# ASP.NET – Web Forms v ASP.NET

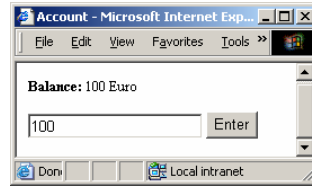


```
<%@ Page Language="C#" Inherits="AdderPage" CodeFile="Adder.aspx.cs"%>
<html>
<head><title>Account</title></head>
<body>
<form Runat="server">
Balance:
<asp:Label ID="total" Text="0" Runat="server"/> Euro

<asp:TextBox ID="amount" Runat="server"/>
<asp:Button ID="ok" Text="Enter" OnClick="ButtonClick" Runat="server" />
</form>
</body>
</html>
```

```
using System;
public partial class AdderPage : System.Web.UI.Page {
public void ButtonClick (object sender, EventArgs e) {
int totalVal = Convert.ToInt32(total.Text);
int amountVal = Convert.ToInt32(amount.Text);
total.Text = (totalVal + amountVal).ToString();
}
}
```

Adder.aspx.cs



Další kapitoly

65

# ASP.NET – Některé ovládací prvky



- Label
- TextBox
- Button
- RadioButton  Radio
- CheckBox  Check
- DropDownList
- ListBox
  - apples
  - pears
  - bananas

Calendar

Febnuar 2003						
Mo	Tu	We	Th	Fr	Sa	Su
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	1	2
3	4	5	6	7	8	9

DataGrid

EmployeeID	FirstName	LastName
1	Nancy	Davolo
2	Andrew	Fuller
3	Janet	Levering
4	Margaret	Peacock
5	Steven	Buchanan
6	Michael	Suyama
7	Robert	King
8	Laura	Callahan
9	Anne	Dodsworth

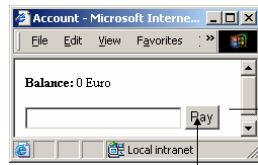
User Controls

Custom Controls

Další kapitoly

66

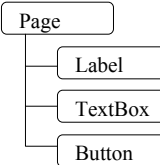
# ASP.NET – Fungování ASP.NET



Click

round trip event

+ page state



## 1. Creation

Je vytvořena stránka a všechny ovládací prvky na ní

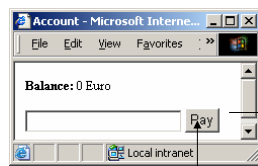
*Client*

*Server*

Další kapitoly

67

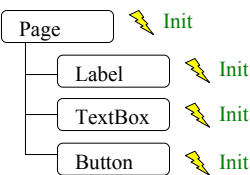
# ASP.NET – Fungování ASP.NET



Click

round trip event

+ page state



## 2. Initialisation

- je vyvolána událost *Init*

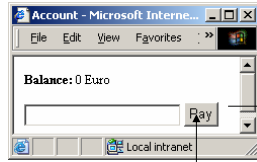
*Client*

*Server*

Další kapitoly

68

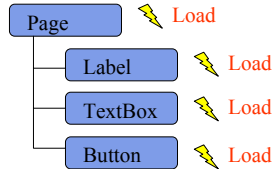
# ASP.NET – Fungování ASP.NET



Click

round trip event

+ page state



### 3. Loading

- load controls with the values that the user has entered (page state)
- raise *Load* events

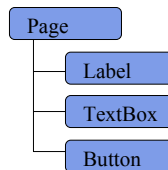
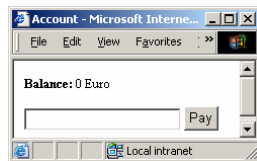
*Client*

*Server*

Další kapitoly

69

# ASP.NET – Fungování ASP.NET



### 4. Action

- handle event(s)  
(Click, TextChanged, ...)

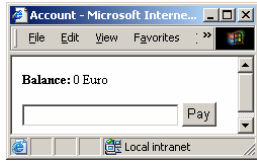
*Client*

*Server*

Další kapitoly

70

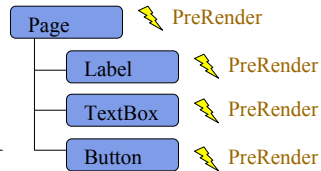
# ASP.NET – Fungování ASP.NET



```
<html>
...
<input type="text" ...>
<input type="button" ...>
...
</html>
```

HTML

+ page state



## 5. Rendering

- raise *PreRender* events
- call *Render* methods of all controls, which render the controls to HTML

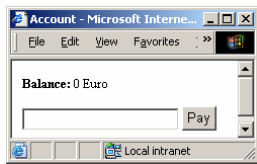
*Client*

*Server*

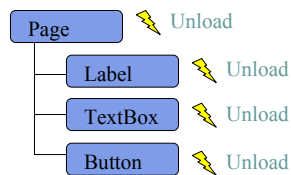
Další kapitoly

71

# ASP.NET – Fungování ASP.NET



```
<html>
...
<input type="text" ...>
<input type="button" ...>
...
</html>
```



## 6. Unloading

- raise *Unload* events for cleanup actions

*Client*

*Server*

Další kapitoly

72