Architektúra a Návrhové vzory platformy .NET

Oliver Krahulec

SOFTEC

Čo je to architektúra?

- Rozdelenie systému na časti?
- Množina návrhov, ktoré sú nemenné?
- Zhodné pochopenie systémového návrhu vývojármi?
- Ucelený plán systému?
- Návrh, ktorý sa mení?
- Kolekcia odporúčaní?
- Túžba dosiahnuť správne riešenie skôr vo fáze návrhu?

SOFTEC

Čo sú to návrhové vzory?

- Riešenie na často sa opakujúce problémy pri návrhu aplikácií
- Pomoc pri vytváraní flexibilných a znovu použiteľných návrhov
- Predstavujú širší rozsah ako jednu triedu alebo algoritmus
- Popisujú okolnosti za ktorých sú návrhy znovu použiteľné
- Popisujú dôsledky a kompromisy ich použitia vzhľadom na širší dizajn aplikácie
- Jazykovo nezávislé
- Používané na budovanie komplexných aplikácií
- Jednoduché vzory sa spájajú a vytvárajú zložitejšie
- Mnohé odporúčania zlepšujú ostatné

SOFTEC

Výhody návrhových vzorov

- Stanovujú slovník
  - Umožňujú tímom komunikovať na vyššej úrovni
  - Znovu použitie technického riešenia dovoľuje zamerať sa na aplikačné problémy
  - Dokumentácia je zrozumiteľnejšia
- Zlepšuje váš kód
  - Lepšie udržiavateľný: Menej prepojení, väčšia súdržnosť
  - Osvojuje spoločné a rozdielne časti
- Zvýšenie produktivity
  - Prečo znovu objavovať koleso?

SOFTEC

Svet návrhových vzorov

- Vzory pre riadenie procesov
- Vzory pre modelovanie organizácií
- Analytické vzory
- Dátové vzory
- Architektonické vzory
- Návrhové vzory
- Implementačné vzory

SOFTEC

Návrhové vzory - GoF

- Autori publikácie sú E. Gamma, R. Helm, R. Johnson a J. Vlissides (Gang of Four)
  - Design Patterns: Elements of Reusable Object-Oriented Software
- Nevynašli vzory, ale prvý krát zdokumentovali a formalizovali dobré myšlienky, ktoré boli používané od začiatku softvérového vývoja.

SOFTEC

Creation patterns

- Vzory sa dotýkajú oblasti optimalizácie vytváraní inštancií.
- Môžu byť rozdelené na vzory typu class-creation a object-creation.
  - class-creation používajú optimálne dedičnosť
  - object-creation používajú delegovanie na vytvorenie
- Abstract Factory
  - Vytvorenie inštancie niekoľkých príbuzných tried
- Builder
  - Oddelenie vytvárania inštancií od ich reprezentácií
- Factory Method
  - Vytvorenie inštancie z niekoľkých derivovaných tried
  - Metóda riadi kedy objekt vznikne, ale konkrétny typ objektu ponechá na podtriedach
  - Factory definuje čo sa má vytvoriť, Builder ako sa to má vytvoriť.
- Prototype
  - Kopírovanie a klonovanie vytvorenej inštancie
- Singleton
  - Trieda, ktorej môže existovať iba jedna inštancia

SOFTEC

Structural patterns

- Dotýka sa skladania Tried a Objektov. Používa dedičnosť na implementáciu rozhraní a definuje spôsob ako skladať objekty na dosiahnutie novej funkcionality.
- Adapter
  - Spája rozhrania rôznych implementácií
- Bridge
  - Oddeľuje objektové rozhranie od implementácie.
- Composite
  - Stromová štruktúra jednoduchých a zložených objektov
- Decorator
  - Dynamické pridávanie zodpovednosti objektom
- Façade
  - Jedna trieda reprezentuje celý podsystém
- Flyweight
  - Optimálne navrhnutá inštancia použitá pre výkonnejšie zdielanie
- Proxy
  - Objekt reprezentujúci iný objekt

SOFTEC

Behavioral patterns

Dotýkajú sa komunikácie medzi objektami / triedami.

- Chain of Resp.
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor

SOFTEC

Architektúra vs. Návrhové vzory

- Vzory v architektúre adresujú problémy
  - Distribuovanej funkcionality a spolupráce systémov
  - Protokolov a Rozhraní
  - Škálovateľnosti, Spoľahlivosti, Bezpečnosti
  - Pôsobia na „Distribuované systémy“
- Návrhové vzory adresujú problémy
  - Udržiavateľnosti, znovu použitia kódu a frameworkov
  - Abstrakcia a ukrývanie algoritmov
  - Pôsobia na “Objektovo orientovaný jazyk”

SOFTEC

Iná cesta programovania

- Poukazuje na objekty čo majú robiť, nie iba ako ich implementovať
- Uvážiť, ktoré sú premenné pri návrhu… a ukryť ich
- Položiť vrstvy medzi časti ktoré sa nezávisle menia

Návrhové vzory = predstavujú najlepšie odporúčania v softvérových návrhoch

- Pomocou metód ukrývame realizáciu, nielen dáta.
- “Stabilita vs. Premenlivosť”

SOFTEC

Charakteristika frameworku

- Čo robí framework framework-om
  - definuje architektúru aplikácie
    - aplikácie majú podobnú štruktúru
    - používateľom sa javia konzistentnejšie
    - sú ľahšie udržiavateľné
    - špecializácia vývojárov
  - Znovu použiteľný dizajn
    - množina abstraktných tried/komponentov spolu s popisom, ako objekty týchto tried/tieto komponenty spolupracujú
  - rýchla prototypizácia aj vývoj
  - prispôsobiteľnosť a rozšíriteľnosť
- Dizajn aplikácie je založený na vzoroch
  - kvalitnejší/overený/lepšie čitateľný kód

SOFTEC

Framework vs. Návrhový vzor

- Návrhový vzor je všeobecnejší, niektorá časť Frameworku môže byť postavená na základe návrhového vzoru
- Framework je teda zostavený spojením tých najlepších odporúčaní z návrhových vzorov
- Framework sa špecializuje na jednu konrétnu oblasť, rieši konkrétnu problematiku, návrhový vzor je v zásade všeobecnejší

.NET Framework

- .NET Framework implementuje množstvo návrhových vzorov
- Súčasťou .NET Frameworku (i C# a ďalších jazykov sú konštrukcie, ktoré nám na infraštruktúrnej úrovni zjednodušujú implementáciu vzorov

Návrhové vzory v .NET

- Observer Pattern
  - Realizovaný pomocou delegátov udalostí
  - Delegát je typovo bezpečný ukazovateľ na funkciu

    ```
    public delegate void ReloadMethod(object sender,
      System.EventArgs args);
    ```
  - Udalosť je špecializovaný delegát s obmedzenou funkčnosťou

    ```
    public event EventHandler Create
    ```
  - Prepojenie na konkrétnu obsluhu udalosti

    ```
    cb_eventCreate.Click += new EventHandler(OnCreate)
    ```
- Factory Pattern
  - Napríklad v triede WebControl
    - Metóda CreateControlStyle v potomkov vytvára konkrétny potomka triedy Style.

Návrhové vzory v .NET

- Page Controller Pattern
  - System.Web.UI.Page – základná časť programového modelu ASP.NET
- Front Controller Pattern
  - Reprezentovaný Http Modulom
  - Implementuje rozhranie

```
interface IHttpModule
{
  Init (HttpApplication context);
  Dispose();
}
```


Návrhové vzory v .NET

- Iterator Pattern
- Decorator Pattern
- Adapter Pattern
- Strategy Pattern
- Composite Pattern in ASP.NET
- Template Method Pattern
- Patterns in the ASP.NET Pipeline
- Intercepting Filter Pattern

SOFTEC

Vrstvy

- Vrstvy reprezentujú zvyšovanie funkcionality
- Poskytujú pripravené bloky pre vyššie vrstvy
- Nižšie vrstvy nikdy nezávisia od vyšších
- Jedna vrstva je samostatným logickým celkom
- Kaskádne zmeny môžu viesť ku zvýšeniu prácnosti
- Zámeny sú možné
- Dodatočné vrstvy môžu znížiť výkonnosť

SOFTEC

Vrstvy

- Tri primárne vrstvy
  - Presentation - Prezentačná
  - Business Logic - Aplikačná
  - Data - Dátová
- Mnohé aplikácie využívajú tieto tri vrstvy

SOFTEC

Web Presentation Patterns

- Model View Controller
  - Ako je možné rozdeliť užívateľské rozhranie a funkcionalitu webových aplikácií aby bolo možné jednoducho modifikovať jednotlivé časti
- Front Controller
  - Ako najlepšie štruktúrovať kontroler pre veľké komplexné aplikácie, aby bolo možné dosiahnuť znovu použiteľnosť a flexibilitu a zároveň sa vyhnúť duplikovaniu kódu
- Page Controller
  - Ako najlepšie štruktúrovať kontroler pre rozumne komplexné aplikácie, aby bolo možné dosiahnuť znovu použiteľnosť a flexibilitu a zároveň sa vyhnúť duplikovaniu kódu

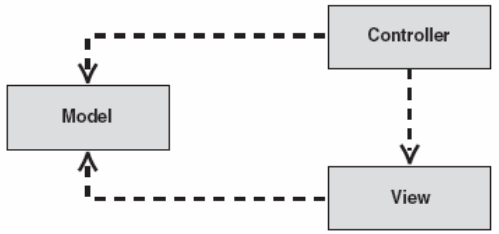
SOFTEC

Model View Controller

- Model.
  Model riadi správanie a dáta v aplikačnej doméne, odpovedá na požiadavky o informácie a ich stave (obvykle z view) a zodpovedá za inštrukcie o zmene stavu (obvykle z controller)
- View.
  View riadi zobrazované informácie.
- Controller.
  Controller interpretuje vstupy od užívateľa, informuje model a/alebo view o potrebných zmenách.

SOFTEC

Model View Controller



SOFTEC

MVC – Jedno - stránková štruktúra

- Model, View and Controller sú na jednom mieste
- Všetky vrstvy v jednom súbore
- Kód vo forme „inline script"
- Jeden súbor nesie aktualizáciu
- Zložitejšia údržba
- Nemožné vytvárať unit testy

SOFTEC

MVC – Jedno - stránková štruktúra

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<html>
   <head>
      <title>start</title>
      <script language="c#" runat="server">
         void Page_Load(object sender, System.EventArgs e)
         {
            String selectCmd = "select * from Recording";

            SqlConnection myConnection =
               new SqlConnection(
                  "server=(local);database=recordings;Trusted_Connection=yes");
            SqlDataAdapter myCommand = new SqlDataAdapter(selectCmd,
               myConnection);
```


SOFTEC

MVC – iba Codebehind

- Oddeľuje view od model a controller
- Všetky vrstvy v jednom súbore
- Kód v triede (základná trieda pre stránku)
- Oddeľuje dizajnéra a programátora
- Jednoduchšia údržba
- Nie je možné vykonávať unit testy

SOFTEC

MVC – iba Codebehind

```
<%@ Page language="c#" Codebehind="Solution.aspx.cs"
    AutoEventWireup="false" Inherits="Solution" %>
<html>
   <head>
      <title>Solution</title>
   </head>
   <body>
      <form id="Solution" method="post" runat="server">
         <h3>Recordings</h3>
         Select a Recording:<br/>
         <asp:dropdownlist id="recordingSelect" runat="server" />
         <asp:button id="submit" runat="server" text="Submit"
            enableviewstate="False" />
         <p/>
         <asp:datagrid id="MyDataGrid" runat="server" width="700"
               backcolor="#ccccff" bordercolor="black" showfooter="false"
               cellpadding="3" cellspacing="0" font-name="Verdana" font-size="8pt"
               headerstyle-backcolor="#aaaadd" enableviewstate="false" />
      </form>
   </body>
</html>
```


SOFTEC

MVC – Codebehind a model

- Všetky časti na oddelených miestach
- Všetky vrstvy v niekoľkých súboroch a knižniciach
- Kód Controller-a v triede (základná „codebehind" trieda ku stránke)
- Oddelenie dizajnéra a programátora
- Jednoduchšia údržba
- Produkuje znovu použiteľný kód
- Unit testy sú možné nad modelom

SOFTEC

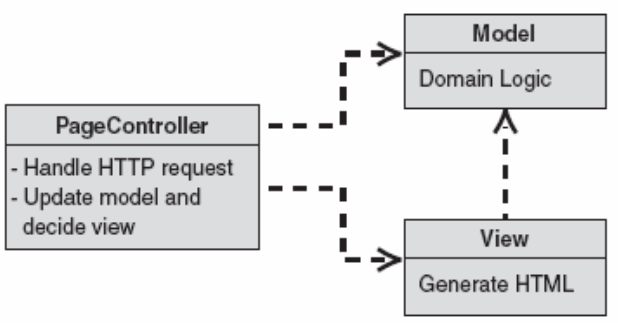
MVC – Codebehind a model

```
using System;
using System.Collections;
using System.Data;
using System.Data.SqlClient;

public class DatabaseGateway
{
   public static DataSet GetRecordings()
   {
      String selectCmd = "select * from Recording";

      SqlConnection myConnection =
         new SqlConnection(
            "server=(local);database=recordings;Trusted_Connection=yes");
      SqlDataAdapter myCommand = new SqlDataAdapter(selectCmd, myConnection);

      DataSet ds = new DataSet();
      myCommand.Fill(ds, "Recording");
      return ds;
   }
```


SOFTEC

Page Controller

- Štandardná implementácia v ASP.NET
- Page Controller je časťou MVC vzoru
- Page Controller určuje pohľad -view



SOFTEC

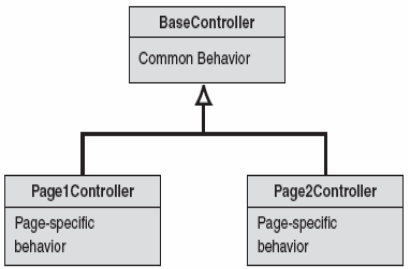
Page Controller

- Životný cyklus stránky (formuláru)
  - ASP.NET inicializácia stránky (Event: Init).
    Inicializuje ASP.NET komponenty podľa objektu Request.
  - Aplikačný kód - inicializácia (Event: Load).
    Keď nastane udalosť **Load**, serverové komponenty sú vytvorené a inicializované, ich stav bol obnovený a formulár odzrkadľuje zmeny klienta. Vytvárajú sa databázové konekcie.
  - Obsluha aplikačných udalostí
    Spracovanie špecifické pre danú aplikáciu ako odpoveď na udalosť vyvolanú controller-om.
  - Vyčistenie (Event: Unload).
    Stránka dokončila všetky renderovania a môže byť zrušená. Sú zrušené všetky nepotrebné objekty a zatvorené konekcie ak to je potrebné.

SOFTEC

Page Controller

- Použiť základnú triedy na definíciu spoločnej funkcionality pre všetky controllery.
- Dediť zo základnej triedy ak vytváram špecifické controllery



SOFTEC

Page Controller

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;

public class Page1 : BasePage
{
   protected System.Web.UI.WebControls.Label pageNumber;

   protected override void PageLoadEvent(object sender, System.EventArgs e)
   {
      pageNumber.Text = "1";
   }
}
```

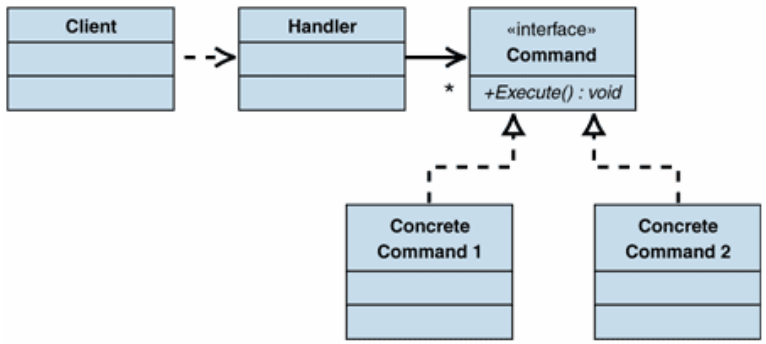

SOFTEC

Front Controller

- Určený pre veľmi komplexné Web aplikácie
- Zachováva znovu použiteľnosť a flexibilitu a vyhýba sa duplikovaniu kódu
- Komplexná navigácia
- URL adresa identifikuje požiadavku

SOFTEC

Front Controller



SOFTEC

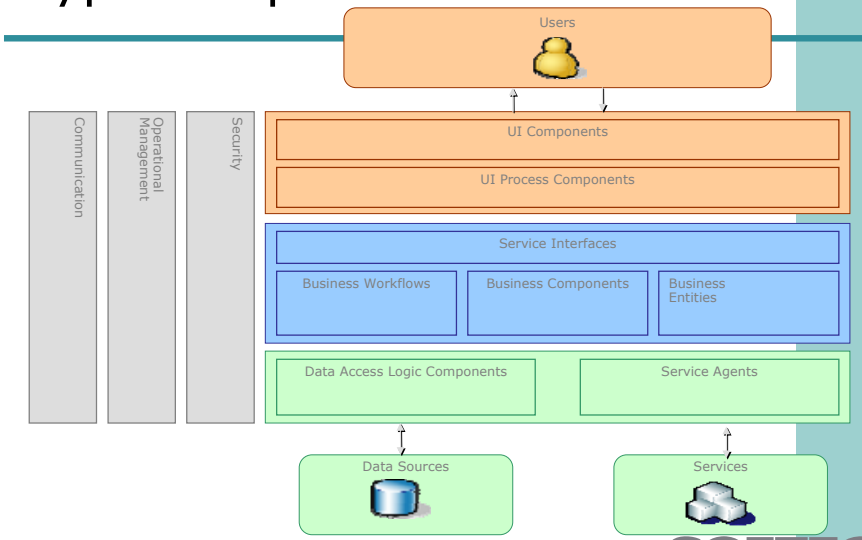
Aplikačné Bloky pre .NET

- Znovu použiteľný kód – C# a VB.NET
  - distribuovaný v zdrojových kódoch
    - možné prispôsobiť, redistribuovať
- Dokumentácia, Quick Start príklady
- *vzory & odporúčania*

  …showcasing best practices
  - Tested & reviewed: Security, Performance, etc.
  - Considering future direction in design
- …lepšie riešenia, rýchlejšie!
- http://msdn.microsoft.com/practices/compcat/default.aspx?pull=/library/en-us/dnpag2/html/entlib.asp

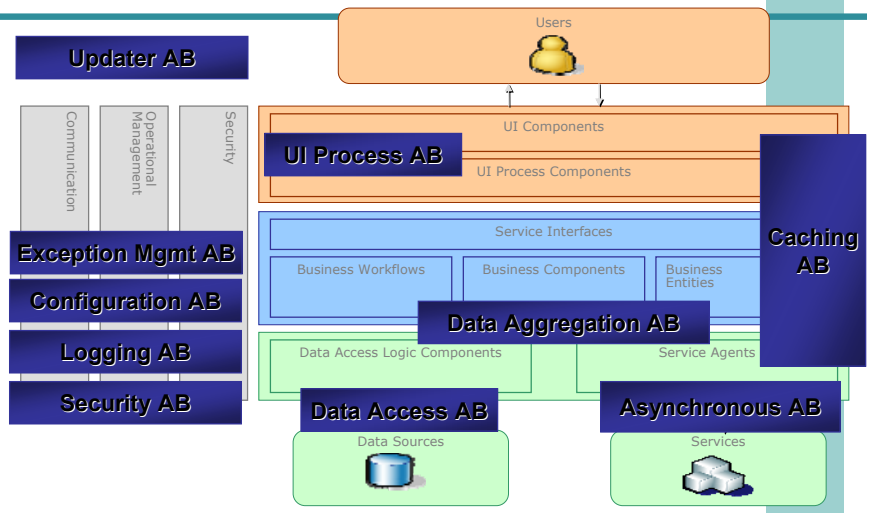
SOFTEC

Typická aplikácia



SOFTEC

Aplikácia Aplikačných Blokov



SOFTEC

Enterprise Library App. Blocks

- Caching Application Block.
  - Umožňuje vývojárom začleniť lokálnu cache do aplikácií.
- Configuration Application Block.
  - Umožňuje čítanie a zápis konfiguračných nastavení.
- Data Access Application Block.
  - Predstavuje komfortnejšiu prácu s externými databázovými systémami.
- Cryptography Application Block.
  - Umožňuje použitie šifrovanie a hashovacích funkcií v aplikáciach.

SOFTEC

Enterprise Library App. Blocks

- Exception Handling Application Block.
  - Umožňuje vytvoriť dôslednú stratégiu pre spracovanie výnimiek, ktoré sa vyskytnú v jednotlivých vrstvách agendových aplikácií.
- Logging and Instrumentation Application Block.
  - Umožňuje použiť štandardné logovanie aplikácií.
- Security Application Block.
  - Umožňuje vývojárom začleniť bezpečnostnú funkčnosť do aplikácií. Aplikácie môžu použiť aplikačné bloky v rôznych situáciách ako sú autorizácia, autentifikácia užívateľov voči databáze, informácie o roliach a profiloch a cache užívateľských profilových informácií.

SOFTEC

Data Access Application Block

- Jeden z najstarších blokov (v.2.)
- Cieľ
  - Zjednodušiť model volania ADO.NET SqlClient
- Čo to je?
  - Implementuje pomocný komponent prístupu k dátam, SqlHelper, ktorý pomáha vykonávať dotazy na SQL Serveri 7.0 a vyššie poskytnutím sady statických metód a redukovaním kódu, ktorý by bolo potrebné napísať. Verzia 2.0 podporuje .NET Framework v1.1.
- Kedy sa používa?
  - Ak pracujeme s databázovým systémom SQL Server a je snaha o redukovanie kódu pre prístup do DB. Môže byť použitý interne, vo vlastných triedach pre prístup k dátam.

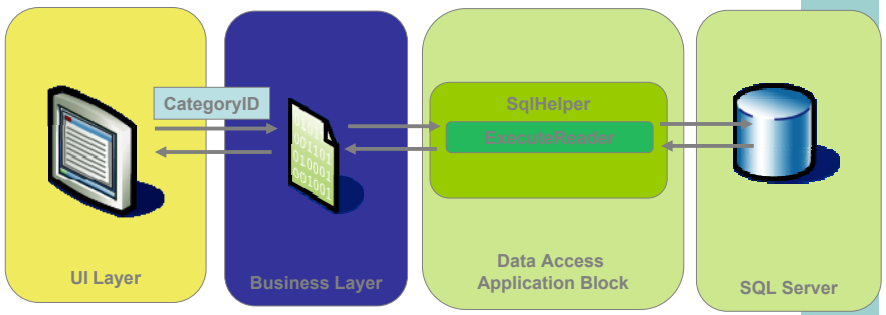
SOFTEC

Data Access Application Block

- Výstup možný vo viacerých formátoch:
  - get DataSets, DataReaders, Scalars, XmlReaders a vykonať dotaz bez vrátenia výsledku – všetko v jednom riadku kódu
- Zjednodušuje volania s viacerými zdrojmi
  - Konekcie, Conn. Strings, SQL transakcie
- Podporuje silne typové DataSet objekty s metódou **FillDataset**
- Podpora potvrdenia zmien v **DataSet** naspäť do databázy
- Doplňujúca podpora pre typové parametre

SOFTEC

Príklad SqlHelper metódy




```
SqlDataReader dr = SqlHelper.ExecuteReader(
          CONN_STRING,"GetProductsByCat", <categoryID>);
```


SOFTEC

Webové služby (Web Services)

- Webové služby sú
  - aplikácie, ktoré umožňujú vzdialené volania po sieti, alebo cez Internet použitím XML a HTTP (TCP/IP)
- Výhody
  - Toto umožňuje ukrytie detailov ako sú služby implementované, potrebná je iba URL a informácia o type dát
  - Je úplne irelevantné pre klienta, či je služba vyvíjaná v Jave alebo ASP.NET, alebo či je prevádzkovaná na platforme Windows, Linux

SOFTEC

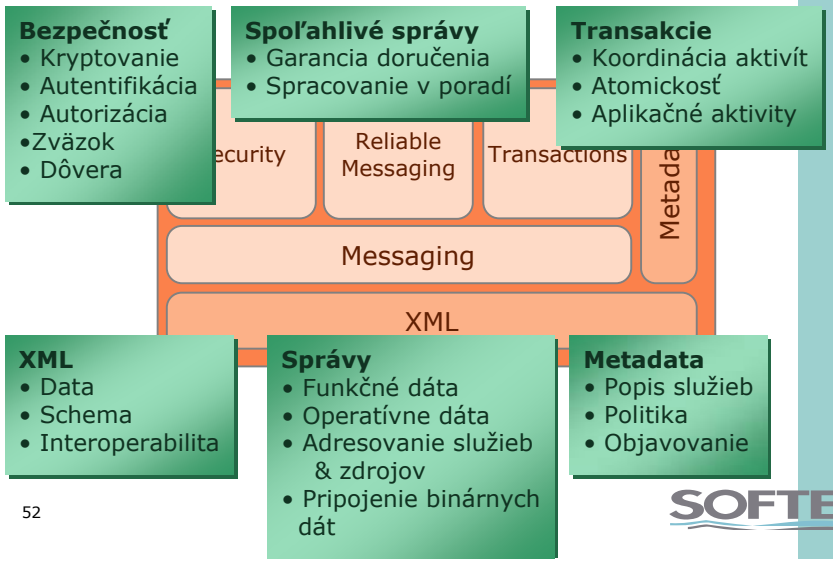
Webové služby (Web Services)

Špecifikácia

- Otvorené štandardné procesy
  - Špecifikáciu navrhujú priemyselní lídri
    - Microsoft, IBM, BEA, atď.
  - Počiatočná implementácia navrhovanej špecifikácie
  - Odozvy a semináre
  - Navrhované špecifikácie predložené štandardným orgánom
    - W3C, IETF, OASIS

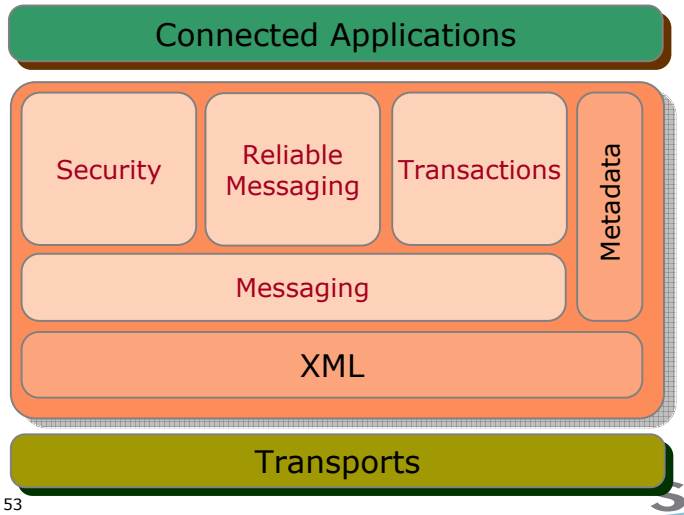
SOFTEC

Web služby - vlastnosti



SOFTEC

Architektúra webových služieb

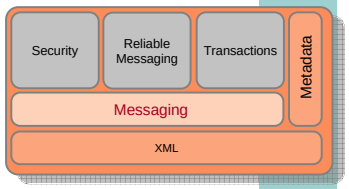


SOFTEC

Architektúra webových služieb

Prehľad

- XML
  - Spoločný výmenný formát dát
- Odosielanie správ
  - SOAP ako jazyk správ
  - Adresovanie správ
  - Pripojenie nie-XML dát do SOAP správy
- Metadata
  - Objavenie služieb
  - Rozhranie služieb popisuje WSDL s XSD
  - Policy popisujú postačujúce požiadavky

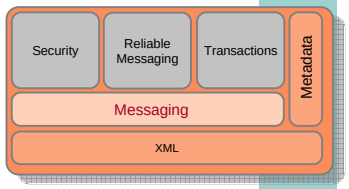


SOFTEC

Architektúra webových služieb

Prehľad

- Bezpečnosť
  - Kritická pre webové služby mimo jednej organizácie
  - Autentifikácia, integrita správ, utajenosť a dôvernosť
  - Federation of security between organizations
- Spoľahlivosť
  - Dôležitá pre úlohy kritických aplikácií
  - Zabezpečenie doručenia správy a spracovania
- Transakcie
  - Ochrana investícií do transakčnej infraštruktúry
  - Rozšírenie na rozličné druhy distribuovaných aktivít



SOFTEC

Architektúra webových služieb

- Návrhové princípy
  - „Skladateľnosť"
  - Modulárnosť
  - Zjednotenosť
  - Spoľahlivosť
  - Decentralizovanosť a autonómnosť
  - Transportná nezávislosť

SOFTEC

„Web Services Enhancements"

- Implementuje množinu vybraných špecifikácií WS architektúry
- Visual Studio .NET Add-in pluginy
- .NET Framework Class Library rozšírenia
- Budované na ASP.NET XML Web Services (ASMX)
- Release plánovanie nezávislé od iných platforiem a nástrojov
- Každý release je plne podporovaný firmou Microsoft

SOFTEC

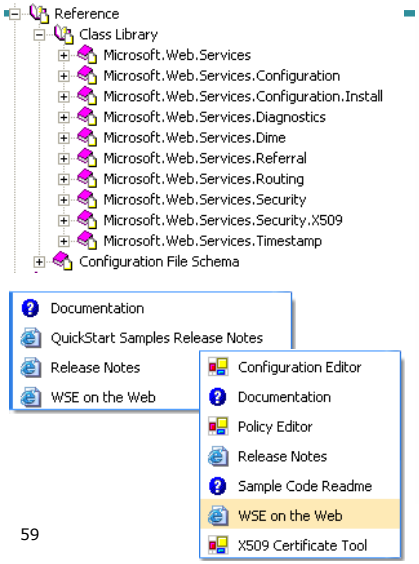
WSE 2.0

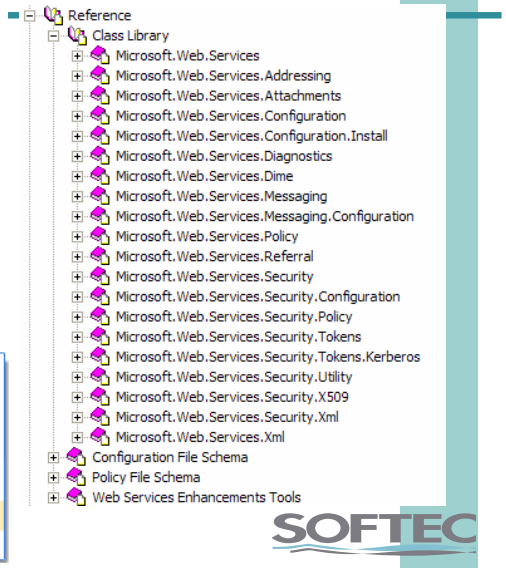
Prehľad

- Prenos
  - HTTP, TCP, in-process
- Správy
  - WS-Adresovanie
  - WS-Prílohy
- Metadata
  - Policy
- Bezpečnosť
  - Dôvernosť, zabezpečená komunikácia, bezpečnostné politiky
  - Kryptografia
    - Kerberos, podpora bezpečnostných XML token-ov
- Nástroje a príklady

SOFTEC

WSE 2.0 – porovnanie s WSE 1.0





SOFTEC

WSE 2.0

- Správy a prenos
  - Prenos
    - Support for HTTP, TCP, in-process
  - Správy
    - WS-Attachments a DIME

      Prílohy pripojené za SOAP obálkou

      Budú nahradené MTOM
    - WS-Addressing

SOFTEC

WSE 2.0

Bezpečnosť

- Autentifikácia
  - Podpora spoločných typov
- Integrita
  - Neodmietnutie: overenie odosielateľa
  - Kontrola obsahu správy
- Dôvernosť
  - Utajenosť
  - Symetrická a asymetrická kryptografia

SOFTEC

WSE 2.0

Bezpečná komunikácia

- Otázka kontextu bezpečnostných tokenov
  - Používanie symetrických kľúčov
  - Nie je potrebné tak veľké množstvo zdrojov na podpísanie a kryptovanie ako pri asymetrických kľúčoch
- Vychádza z WSE 1.0

SOFTEC

WSE 2.0

Prílohy

- Dáta ktoré je ťažko serializovať
  - Binárne dáta
  - Šifrované dáta
  - Veľké XML dokumenty
- DIME - Direct Internet Message Encapsulation
  - Prílohy pripojené za SOAP obálkou
  - SOAP obálka je dostupná
  - WS-Attachements a DIME budú nahradené MTOM

SOFTEC

Servisná orientácia

- Servisne orientovaná architektúra (SOA) vo významnej miere zvyšuje úroveň abstrakcie znovu použiteľnosti kódu
  - Umožňuje aplikáciám spojenie so službami, ktoré sa v čase rozvíjajú a zlepšujú, bez potrebných modifikácií aplikácii, ktoré ich používajú
- Evolúcia, nie revolúcia !

SOFTEC

Služby a systémy

- Služba je program s ktorým komunikujete pomocou výmeny správ
  - Služby sú vytvorené nakoniec
  - Dostupnosť a stabilita sú kritické časti
- Systém je množina nasadených služieb spolupracujúca na danej úlohe
  - Systémy sú budované pre výmenu
  - Adaptácia na nové služby po nasadení

SOFTEC

Čo je Servisná Orientácia (SO)?

- Prístup pri tvorbe systémov použitím služieb dodržuje tieto 4 princípy Servisnej Orientácie:
  1. Ohraničenie je explicitné
  2. Služby sú autonómne
  3. Služby zdieľajú schémy, nie triedy
  4. Kompatibilita služieb závisí od politiky

SOFTEC

Porozumenie pravidla č.1

Ohraničenie je dané

- Každé interakcia medzi službami je ohraničená
- Prekročenie ohraničenia služieb môže byť nákladné
- Servisná orientácia vytvára interakciu formálnu, úmyselnú a explicitne jasnú

SOFTEC

Porozumenie pravidla č.2

Služby sú autonómne

- Topológia systémov sa v čase vyvíja
- Nepresadzuje sa autorita
- Služby v systémoch sú nasadzované, spravované, verziované nezávisle
- Služby nemôžu zlyhať

SOFTEC

Porozumenie pravidla č.3

Služby zdieľajú schému, nie triedu

- Služby spolupracujú výhradne pomocou schém pre štruktúry a kontraktu pre správanie
- Na rozdiel od OO tried, služby nespájajú štruktúry a správanie
- Počítačová overiteľnosť nám umožňuje ochrániť integritu služby
- Kontrakt a schéma zostávajú nemenné po celý čas

SOFTEC

Porozumenie pravidla č.4

Kompatibilita služieb je založená na policy

- Oddelenie interakcií služby, ktoré služba môže mať od obmedzení pri interakcií
- Schopnosti a požiadavky služieb sú vyjadrené vo výrazoch policy
- Uplatnenie – deklarácia výrazu policy je identifikovaná stabilným, globálne unikátnym menom, ktoré nezavisí od miesta ani času

SOFTEC

Posunutie k servisnej orientácií

Od

- Spojenie = náklady
- Funkčne orientované
- Vytvárané ako posledné
- Dlhotrvajúci vývoj

Do

- Spojenie = hodnota
- Procesne orientovaní
- Vytvárané pre výmenu
- Prírastkovo nasadzované

- Ťažko previazané
- Objektovo orientované

- Voľne previazané
- Orientované na správy

SOFTEC

Servisná orientácia

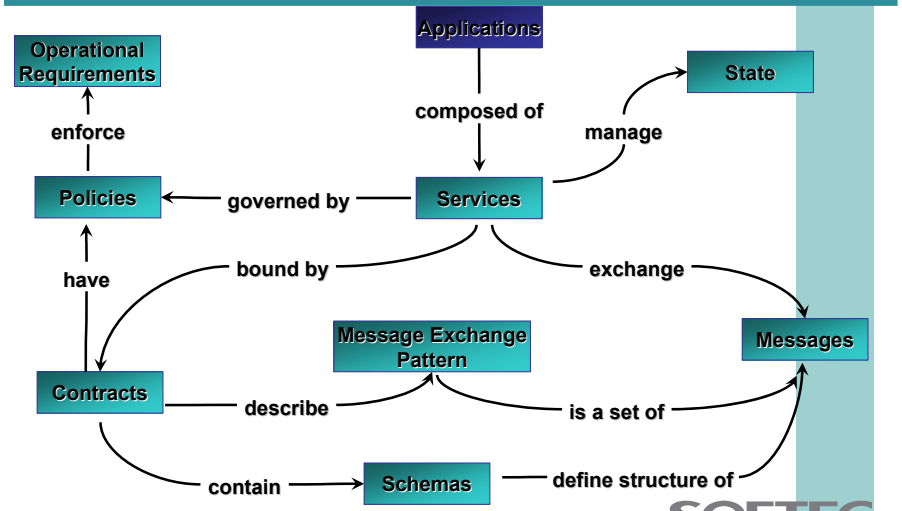
Požadované schopnosti

- Bezpečnosť - Security
- Spoľahlivosť - Reliability
- Transakčnosť - Transactions
- Možnosť „objavania" - Discovery
- Riadenie - Management
- Prenosová nezávislosť - Transport independence
- Interoperability
- Orchestrácia procesov - Process orchestration

SOFTEC

Servisná orientácia

Kľúčové koncepty



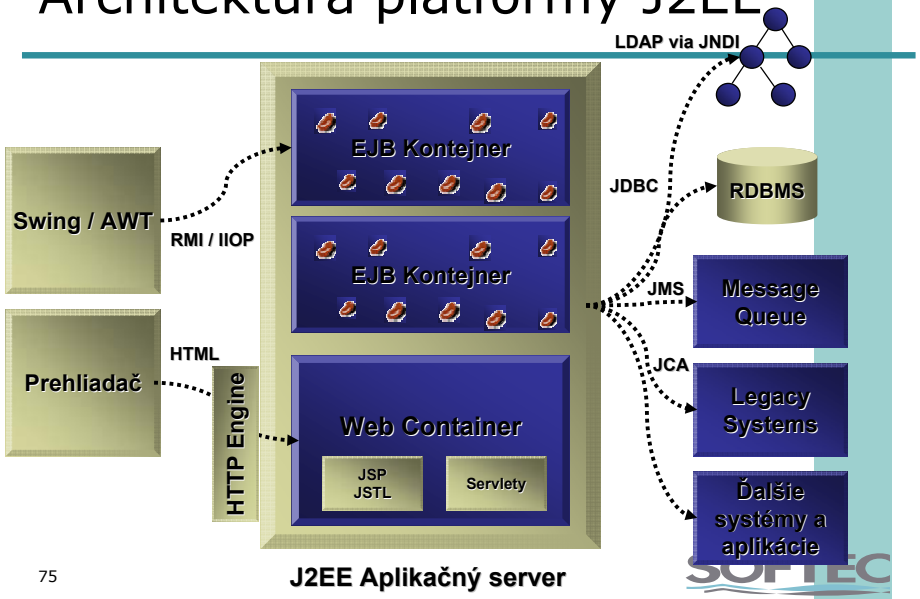
SOFTEC

Úvodné porovnanie .NET a Java

- Podobnosti
  - Veľmi dobre štruktúrované knižnice
  - Kód je kompilovaný do medzikódu
- Rozdielna filozofia
  - Java – platforma je určená pre jeden programovací jazyk, snaží sa o vytvorenie prieniku funkcionality cez OS
  - .NET – platforma je určená pre viacej programovacích jazykov, i keď je navrhnutá platformovo nezávisle, snaží sa „vytiahnuť" z OS maximum funkčnosti
- Výhody .NET
  - Jednoduché a rýchle použitie, minimalizácia kódu
  - Bezpečnosť a integrácia s OS
  - Verziovanie a správa

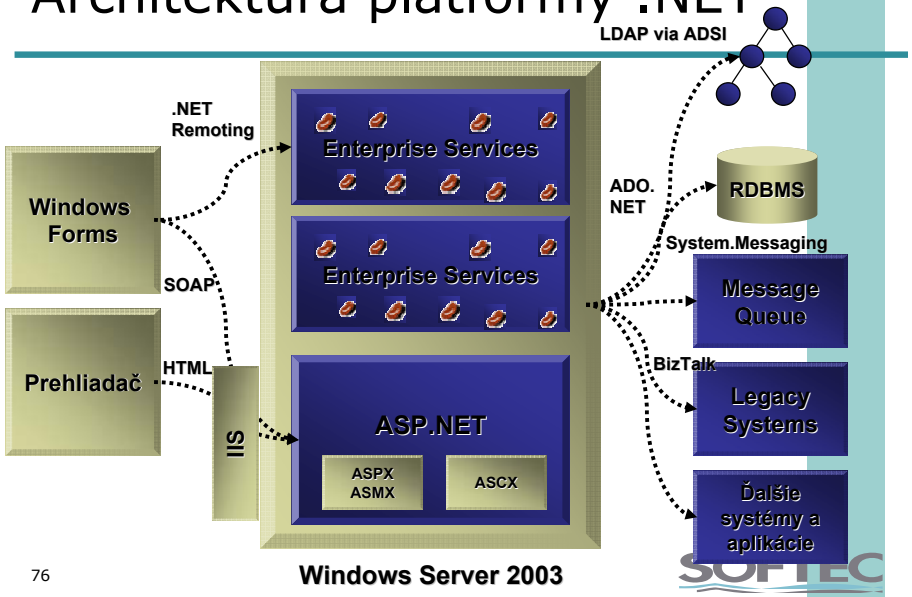
SOFTEC

Architektúra platformy J2EE



SOFTEC

Architektúra platformy .NET



„Tučný klient"

Java	.NET
Swing	Windows Forms
AWT	
Ďalšie (WFC, SWT)	
Optimalizácia cez JNI alebo iný produkt (napr. JACOB)	Sú priamo integrované do systémového GUI => rýchla odozva, možnosť používať všetky GUI prvky systému

Niektoré komponenty	Niektoré komponenty
javax.swing.JFrame	System.Windows.Forms.Form
javax.swing.JButton	System.Windows.Forms.Button
javax.swing.JTextField	System.Windows.Forms.TextBox

„Tučný klient" - Java a C#

```
//Java
class SimpleFrame extends DecoratedFrame {
  Button buttonOK = new Button();
  Button buttonCancel = new Button();
  void buttonOK_actionPerformed(ActionEvent e) {
    System.out.println("OK clicked");
  }
  void buttonCancel_actionPerformed(ActionEvent e) {
    System.out.println("Cancel clicked");
  }
  void jbInit() throws Exception {
    buttonOK.setLabel("OK");
    buttonOK.addActionListener( new ActionListener() {
       public void actionPerformed(ActionEvent e) {
         buttonOK_actionPerformed(e);
       }
    });
    buttonCancel.setLabel("Cancel");
    buttonCancel.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
         buttonCancel_actionPerformed(e);
       }
    });
    this.add(buttonOK, BorderLayout.NORTH);
    this.add(buttonCancel, BorderLayout.SOUTH);
  }
}
```

```
//C#
class SimpleForm extends Form
{
  Button buttonOK = new Button();
  Button buttonCancel = new Button();
  void buttonOK_click(Object sender, Event e)
  {
    System.out.println("Clicked OK");
  }
  void buttonCancel_click(Object sender, Event e)
  {
    System.out.println("Clicked Cancel");
  }
  void initForm()
  {
    buttonOK.setText("OK");
    buttonOK.addOnClick(new EventHandler(this.buttonOK_click));
    buttonCancel.setText("Cancel");
    buttonCancel.addOnClick(
       new EventHandler(this.buttonCancel_click));
    this.setNewControls(new Control[] {buttonCancel, buttonOK});
  }
}
```


Komponentové technológie

Java	.NET
AWT/JavaBeans event model	.NET event and delegate objects
Swing event model	
JavaBeans properties (get/set)	.NET properties (prístupné priamo)
XDoclet a javadoc tagy	.NET atribúty

Nasadenie	Java	.NET
Packaging	Package (jar)	Assembly (dll, exe, moduly)
Umiestnenie	Classpath	GAC, aplikačná politika
Verziovanie	N/A	Strong name

Webové technológie

Java Host	.NET Host
Servlet kontejner, napr. Apache Tomcat	**IIS**

Java Web komponenty	.NET Web komponenty
JavaServer Pages	**ASP.NET**
Java Servlet	**HttpHandler, HttpModule**
Tag Libraries	**ASP.NET server controls**
Applet	**Windows Forms user control umiestnené v ASP.NET Web form alebo J# Browser control**

JSP a ASP.NET

JSP	ASP.NET
Vychádzajú z modelu Microsoftu ASP, rozširujú ich o Javabeans	Rozširuje model JSP o úplné oddelenie kódu a zobrazenia
Veľmi dobrý výkon (ale pre optimalizované aplikácie sa nasadzujú servlety)	Veľmi dobrý výkon
JSF	Web Forms, Server Controls, Custom Controls
„Nepopulárne" postupy ako Taglibrary	Veľmi ľahké použitie, v VS.NET je možné prepínať medzi kódom na pozadí a zobrazením

SOFTEC

Aplikačná infraštruktúra

Java Service	.NET Service
J2EE Servlet Container	**IIS**
J2EE EJB Container	**Enterprise Services**
Java Transaction Service (JTS)	**Enterprise Services a Distributed Transactional Coordinator (DTC)**
Java Transaction API (JTA)	**Enterprise Services Bring Your Own Transactions (BYOT)**
Java Messaging Service (JMS)	**System.Messaging namespace a MSMQ**
Java Naming and Directory Interface (JNDI)	**Active Directory Service Interface (ADSI)**
(RMI)	**.NET Remoting**

SOFTEC

Enterprise Services

- Poskytujú podporu pre
  - Distribuované transakcie
  - Bezpečnost
  - Object Pooling
  - Just in time aktivácia
  - Queued Components
  - Loosely Coupled Events
  - Server Application Process Model
- Nasledovník MTS a COM+

SOFTEC

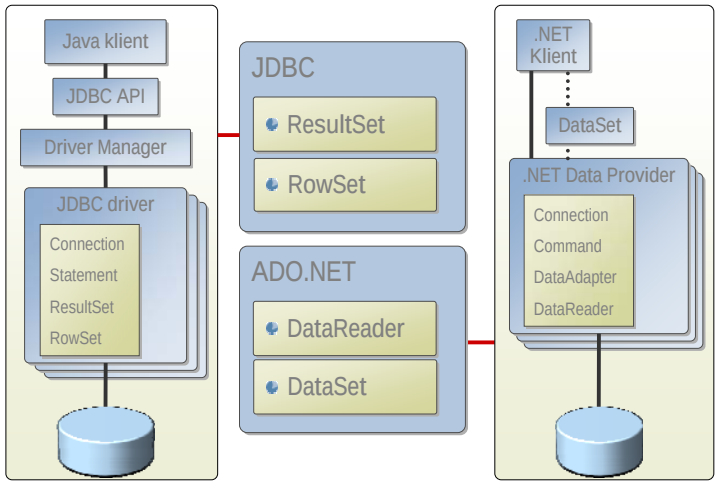
Riadenie bezpečnosti

- Bezpečnosť Java aplikácií je spravovaná Java Authentication and Authorization Service (JAAS)
- Bezpečnosť .NET security je spravovaná Runtime Security Policy

Java Security	.NET Security
JAAS	**Role-based security**

SOFTEC

Porovnanie ADO.NET a JDBC



SOFTEC

Základné ciele ADO.NET

- Kompatibilné s ADO
- Všadeprítomná podpora XML
- Orientácia na dáta, odpojené od DB
- Komplexný objektový model (serverové dáta, XML, offline dáta atď.)
- Maximálny výkon
- Super rýchly s SQL serverom
- Ideálny pre heterogénne prostredie

SOFTEC

	.NET	Java
GUI	WinForms (AWT on J#)	SWING / AWT
Web prostredie	ASP.NET	JSP, JSF
Web GUI (pôvodné)	ASP	JSP
Web controls	Web Forms, Server a Custom Controls	JavaServer Faces, Custom tags
Web Scripting	HttpHandler, HttpModule	Servlet, Filter
Server Side BL Component	Serviced Component (COM+)	EJB Session Beans
Server Side Data Component	Serviced Component (COM+) with DB Logic	EJB BMP Entity Beans
Server Side Data Component	Object Spaces (pozastavené)	Čiastočne EJB CMP Entity Beans
Naming	ADSI	JNDI
Remote Invocation	.NET Remoting	RMI / RMI-IIOP
Data Access	ADO.NET	JDBC3.0
Data Access (pôvodný)	ADO, ODBC	JDBC, SQL/J
Messaging	MSMQ	JMS
Transactions	COM+ / MTS	JTA

SOFTEC

Základné nástroje v devkit

Java	.NET
javac	**Kompilátory .NET—csc, vjc, vbc, ...**
javadoc	**csc s parametrom /doc**
IDEs such as Eclipse, IDEA...	**Visual Studio.NET, Borland C# Builder, ...**
jar	**Assemblies sú vytvárané odpovedajúcim kompilátorom**
java.exe	**Aplikácie sú kompilované ako .exe súbory. CLR je potom automaticky vyvolané systémom**

SOFTEC

10 dôvodov pre ASP.NET

1. Visual Studio .NET
2. Omnoho jednoduchší model vývoja
3. Oddelenie kódu od vzhľadu a obsahu
4. Kompilované jazyky (VB.NET, C#, …)
5. Aktualizácia bez výpadkov (XCOPY)
6. Podpora mobilných zariadení
7. Spoľahlivé uloženie stavu (i pre farmy)
8. Modulárna, plne rozšíriteľná architektúra
9. Automatická detekcia chyb so zotavením
10. Veľmi, veľmi rýchle

SOFTEC

Záver

- Integrácia platforiem .NET a J2EE
- Kombinácia J2EE a .NET s využitím jeho výhod
- Budúcnosťou sú odborníci so znalosťou oboch platforiem
- Neviazanosť s jedným dodávateľom

SOFTEC

Otázky?

SOFTEC