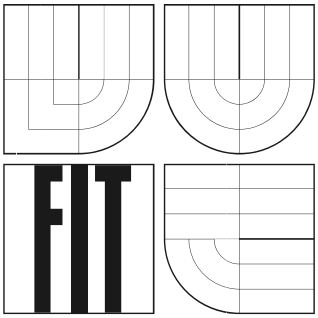
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



# Informační systém elektronické aukce

Ročníkový projekt

Brno 2006 Martin Maslaňák

# Informační systém elektronické aukce



## Prohlášení

Prohlašuji, že jsem tuto ročníkovou práci vypracoval samostatně pod vedením ing. Petra Blatného.
Další informace mi poskytl Radim Burget.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum:

# Abstrakt

Tento ročníkový projekt se zabývá technikou tvorby dynamických webových stránek a webových aplikací pomoci Java Server Pages. Nejprve je potřeba objasnit co to dynamické stránky jsou a jaké techniky se využívali dříve. Poté následuje srovnání JSP s obdobnými prostředky s využitím Active Server Pages .NET. Další část patří tvorbě systému online elektronické aukce s využitím frameworku Struts. Seznámení s tímto frameworkem a nastínění výhod a usnadnění, které práce pomoci Struts poskytuje.

# Klíčová slova

Java Server Pages, Active Server Pages, J2EE, .NET, dynamické webové stránky, Struts, Apache Tomcat, Eclipse, Servlet, MVC.

# Poděkování

Chtěl bych poděkovat Jiřímu Ševčíkovy za pomoc při návrhu designu stránek.

# Abstract

This project deal with a technique of doing dynamic web pages a web applications by the help of Java Server Pages. In the first part of this work I am making clear what the dynamic web pages are about and which techniques were used earlier. After it I compare Java Server Pages with a similar methods in Active Server Pages .NET. In the last part of the documentation I am explaining how to make system of an online elektonic auction by using framework Struts, identification framework Struts and adumbration advantages, which work with this framework provides.



# Keywords

Java Server Pages, Active Server Pages, J2EE, .NET, dynamic web pages, Struts, Apache Tomcat, Eclipse, Servlet, MVC.

# Obsah

# Úvod

Programování vícevrstvých aplikací je rozsáhlou oblastí, ve které se lze setkat s řadou různých technologií, aplikačních serverů , nástrojů a postupů.

Úkolem mé ročníkové práce je porovnání metod tvorby dynamických webových stránek pomoci dvou rozdílných technologií. První je tvorba stránek s využitím Java Server Pages (JSP) od Javy a druhá je pomoci konkurenčního Active Server Pages (ASP) .NET od firmy Microsoft. Další část ročníkové práce je samotná tvorba aplikace internetové aukce s využitím prostředků JSP.

Pro zjištění rozdílů mezi těmito metodami je nutné se nejdříve seznámit s tvorbou dynamických webových stránek, tj. pochopit, jak se liší tyto stránky od statických. Rozdíl takových stránek je v tom, že statické stránky nejsou schopny měnit vzhled po nahrání do prohlížeče, tzn. pokud by uživatel potřeboval jinou, byť jen nepatrně změněnou stránku, musí být ze serveru nahrána úplně jiná stránka.

Při tvorbě aplikace jsem využil frameworku Struts od organizace Apache Software Foundation, který popíši v další části dokumentace.

# 1 Dynamické webové stránky

Dají se rozdělit mezi dvě kategorie.

## 1.1 Dynamika na straně klienta

Tyto typické statické stránky bylo možno rozšířit o jistou interaktivitu začleněním ActiveX ovládacích prvků nebo Java appletů. Nicméně např. ActiveX prvky je nutné na počítač klienta nejdříve nahrát a zaregistrovat v registru, což je nepohodlné a často i nebezpečné, není-li zdroj těchto prvků spolehlivý. Proto došlo k vytvoření specifikace DHTML.

Za dynamické stránky lze považovat stránky, vyhovující specifikaci DHTML (Dynamic Hypertext Markup Language). Pod touto zkratkou se nerozumí žádný nový jazyk, jedná se o spolupráci klasického HTML se skripty, spouštěnými na straně klienta (nejčastěji JScript, méně pak VBScript), CSS (Cascading Style Sheets) a řady dalších technologií, které jsou společně propojeny objektovým modelem DOM (Document Object Model), schváleným W3C (World Wide Web Consorcium) 1.října 1998.

## 1.2 Dynamika na straně serveru

Pod dynamičností stránky lze rozumět rovněž schopnost přizpůsobit svůj vzhled parametrům, které získá od klienta, již na serveru, který stránku dynamicky podle těchto kritérií vygeneruje. V praxi jde např. o situaci, kdy se uživatel registruje do nějaké databáze a posléze obdrží stránku se zadanými parametry (třeba pro kontrolu a případnou změnu zadaných informací). Po shlédnutí zdrojového kódu bychom zjistili, že příslušná pole jsou běžně vyplněna např. jménem nebo příjmením registrovaného (nikoli skripty, které by identifikovaly pozici právě přidaného záznamu v databázi). Není to tím, že by pro každého registrovaného uživatele byla předpřipravena nová stránka, protože nevíme, kolik uživatelů se bude registrovat. V tomto případě mluvíme o skriptech, vykonávaných na straně serveru. Klientovi je vždy zaslán pouze výsledek těchto skriptů, nemá tedy možnost přistupovat k jejich zdrojovému kódu. Pro interakci mezi serverovým skriptem a klientem je nutné nějak k serveru přenést uživatelem zadané informace. Tyto parametry jsou často součástí adresy webové stránky (např.

http://www.neco.cz/stranka.asp?jmeno=Alena&prijmeni=Novakova). Je však rovněž možné je umístit do hlavičky požadavku HTTP, takže zůstanou běžnému uživateli skryté.

Technologie, používané pro vygenerování stránek na serveru, lze rozdělit do dvou kategorií:

*1. Skriptové vsuvky*

V tomto případě se kód skriptu kombinuje přímo se zápisem webové stránky. Když je server vyzván k odeslání této stránky klientovi, nejprve ji celou projde, serverové skripty vykoná (jejich výstupem musí být HTML kód (resp. XML aj.)) a výsledek odešle klientovi.

Tato filozofie dynamicky generovaných stránek v současnosti převažuje. Dnes tímto způsobem fungují téměř všechny používané technologie (SSI, ASP, ASP.NET, PHP, JSP, ColdFusion)

*2. CGI – skripty*

Tyto skripty jsou napsány v libovolném programovacím jazyce a zkompilovány do nějakého spustitelného souboru (např. exe). Pokud klient skript volá, volá vlastně spustitelný soubor, který od něj převezme parametry, na základě nichž vytvoří webovou stránku. K psaní CGI-skriptů se nejčastěji používá jazyk Perl, nicméně lze bez problémů použít např. C/C++, Javu a další.

# 2 Technologie pro generování stránek na straně serveru

Nyní bude následovat stručný přehled technologií, v další části dokumentace se potom budu detailněji zabývat technologiemi JSP a ASP .NET.

## 2.1 Hypertext preprocesor (PHP)

PHP je skriptovací jazyk, který se přímo začleňuje do textu HTML. Historie PHP sahá do roku 1994, kdy se Rasmus Lerdorf rozhodl naprogramovat v Perlu množinu skriptů sloužící k evidenci přístupu k jeho stránkám. Později Lerdorf systém portoval do jazyka C. Nedlouho po uvolnění úvodní verze PHP začalo PHP používat stále více lidí. Zanedlouho Lerdorf přidal ještě program Form Interpreter *(FI)*, který zpřístupňoval databázové systémy na webu. Brzy se stalo PHP celosvětově populární.

Funkce PHP se velmi podobá ASP. Pokud webový server zjistí, že požadavek HTTP směřuje na PHP stránku, je dotaz přesměrován do modulu PHP ISAPI, který celou stránku zkompiluje. Poté je výsledek předán zpět webovému serveru, který jej odešle klientovi.

PHP za svou popularitu vděčí nezávislosti na platformě (lze jej přenášet mezi Windows, Unix, Mac OS, OS/2), jednoduchosti syntaxe a stabilnosti. K vývoji PHP webu lze použít velké množství open – source programů.

## 2.2 Server side include (SSI)

Jde o nejstarší a nejrozšířenější druh vkládaných vsuvek. Použití je zcela jednoduché: do stránky stačí přidat zápis ve tvaru `<!--#příkaz parametr="hodnota"-->` a server tuto značku poznámky nahradí za příslušný textový řetězec, podporuje-li SSI. V opačném případě se řetězec nevyhodnotí a klientovi je odeslána poznámka, která se nezobrazí.
Na stránce tak můžeme zobrazit např. velikost příslušného dokumentu, datum jeho poslední modifikace, datum a čas na serveru apod.

## 2.3 Active Server Pages

ASP je předchůdce ASP .NETu. Je to v podstatě filtr ISAPI – knihovna DLL, která se nahrává do stejného paměťového prostoru jako webový server v okamžiku, kdy je poprvé požadována. Výhodou je, že i při vzájemném přístupu více klientů odpadá zatěžování serveru nahráváním stále nových instancí.

Tato technologie byla uvedena roku 1996 firmou Microsoft. Její kódové jméno bylo Denali. ASP je uložena v jediné knihovně s názvem ASP.DLL (asi 300 KB). Kdykoli klient požaduje stránku .asp, předá webový server jeho žádost filtru ISAPI ASP, který se postará o vykonání všech skriptů. Výsledek ASP je kód HTML (nebo XML aj.), jenž se vloží do textového toku, odeslaného klientovi.

## 2.4 Active Server Pages .NET

Po uvedení komplexní síťové platformy .NET od Microsoftu v únoru 2002 byla rovněž představena nová verze ASP. Je to nově vybudovaná technologie nad .NET frameworkem, která využívá veškeré výhody, které .NET nabízí.

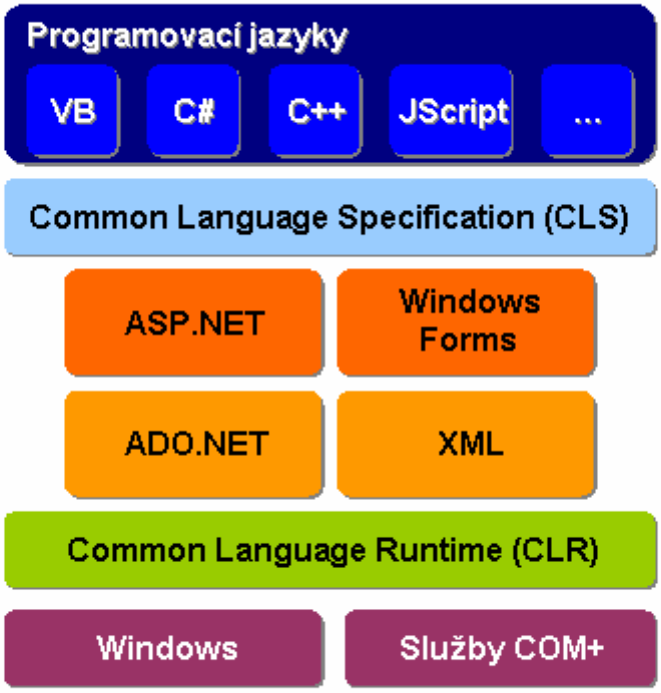
## 2.5 Java Server Pages

Roku 1997 byly firmou Sun Microsystems představeny malé programy, generující webové stránky na straně serveru - servlety. O rok později Sun představuje komplexní platformu J2EE, na jejímž základě jsou postaveny i produkty jiných softwarových společností. Roku 1999 se Sun nechal inspirovat technologií ASP a uvádí specifikaci JSP, které je postavena platformě J2EE. **[1]**

# 3 Active Server Pages .NET

## 3.1 Popis .NETu

Pro činnost webových stránek v ASP.NET je třeba komponenta zvaná Microsoft .NET Framework. .NET se postará o řadu nízkoúrovňových a nezáživných povinností jakými jsou:




**Obrázek 3-1:** .NET framework

- ADO.NET – knihovna pro práci s daty s možností jejich XML reprezentace
- Windows Forms – knihovna pro vývoj uživatelského rozhraní pro desktopové aplikace
- ASP.NET – knihovna pro vývoj uživatelského rozhraní pro webové aplikace.

Základem koncepce .NET je Common Language Runtime (CLR), což je prováděcí prostředí pro všechny .NET aplikace. Kód, který je spuštěn v rámci CLR, se nazývá managed code, což znamená,

že aplikace vyvinutá v libovolném programovacím jazyce či prostředí se musí řídit podle specifikací určených v .NET. (Jsou vlastně jen dvě, CLS neboli Common Language Specification, určující povahu vývojového jazyka, a CTS neboli Common Type System, určující předávání hodnot, typy atd.). Takové aplikace jsou plně pod správou CLR, které odpovídá za jejich správný běh. Kromě toho je ale možné spustit i takzvaný unmanaged code, což je kód, který se neřídí těmito specifikacemi, jako například COM komponenty.

Činnost CLR lze popsat poměrně krátce v několika bodech:

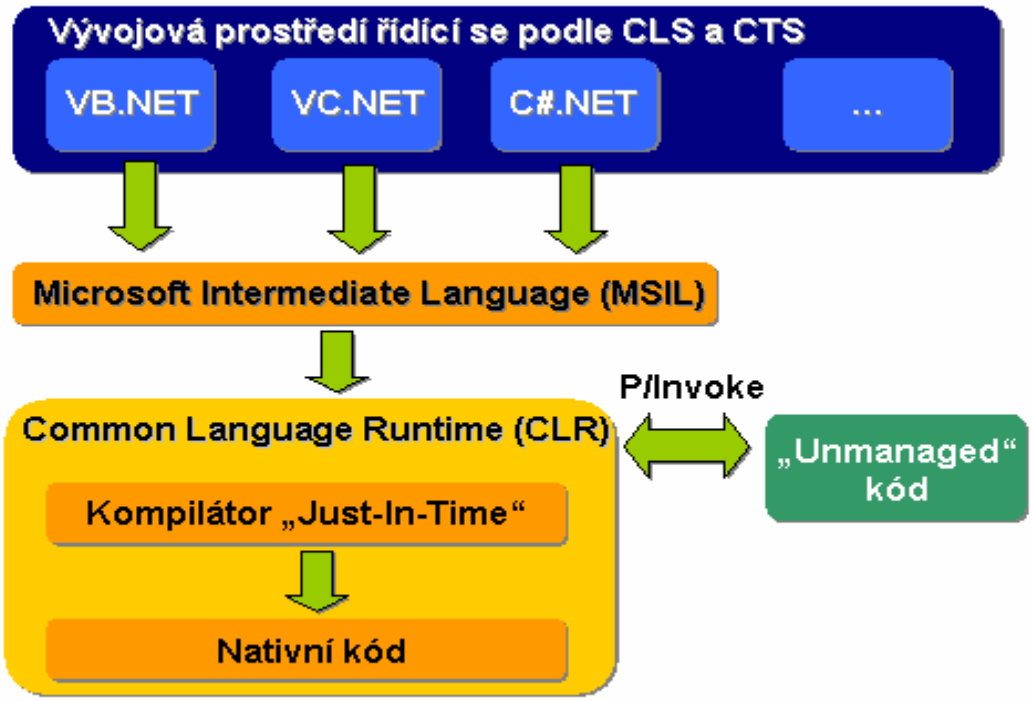
- Správa kódu, který běží v rámci CLR, kontrola typové správnosti, správa paměti, „garbage collector“, zachytávání výjimek.
- Poskytuje přístup k systémovým službám v rámci Windows API nebo přístup ke COM službám.
- Poskytuje služby propojení aplikací vyvinutých v různých programovacích jazycích.

Velmi důležitou částí .NET frameworku jsou podporované jazyky. .NET framework je jazykově nezávislý, pro libovolnou úlohu lze v principu použít jakýkoliv z podporovaných jazyků. Pro psaní webových aplikací jsou nejlepší C# a Visual Basic.NET.

Výsledek těchto dvou jazyků bude stejný, pokud jde o funkčnost i výkonnost. Visual Basic.NET připomíná skriptovací jazyky. C# je elegantní, moderní, sevřený, ale obtížněji čitelný než Visual Basic .NET.

## 3.2 Integrace vývojových prostředí

.NET je otevřen v podstatě libovolnému vývojovému nástroji nebo programovacímu jazyku, který dokáže svůj kód kompilovat do MSIL (Microsoft Intermediate Language).




**Obrázek 3-2:** Integrace vývojových prostředí

Microsoft v tomto ohledu nabízí své Visual Studio .NET, které je možné pokládat za špičku v oboru vývojových nástrojů a které je s .NET prostředím zcela integrováno. Kromě toho mohou i jiní programátoři zůstat věrní svým nástrojům, pokud dodavatel jejich vývojového prostředí vytvořil kompilátor do MSIL kódu. **[2]**

## 3.3 Active Server Pages .NET

ASP .NET je reakce firmy Microsoft na konkurenční technologie, převážně pak JSP běžící na platformě J2EE (viz. dále) od firmy SUN. Společného s předchozí technologií ASP má převážně jméno. Princip komunikace mezi klientem a serverem je sice podobný, tzn. Požadavek na zobrazení webu je zaslán na server, ten vygeneruje HTML kód, který pošle zpět uživateli. ASP .NET je založen na CLR a je sdílen všemi aplikacemi postavenými na frameworku .NET. Aplikace založené na ASP .NET jsou rychlejší, protože jsou předkompilovány do jednoho nebo několika málo DLL souborů, na rozdíl od ryze skriptovacích jazyků (ASP, PHP), kde jsou stránky při každém přístupu znovu parsovány.

Stránky jsou poskládány z objektů, ovládacích prvků. Při tvorbě je možné použít ovládací prvky, kterých je opravdu velké množství (tlačítka, labely, textová pole aj.). Těmto prvkům lze pak přiřazovat určité vlastnosti a zachytávat na nich události. Prvky produkují HTML kód, který tvoří část výsledné stránky poslané do uživatelova prohlížeče. Velkou výhodou je také instalace projektů a aplikací.

Jelikož .NET framework je tvořen jako ucelený prostředek pro vývoj nejen webových aplikací, obsahuje i výbornou softwarovou podporu (Windows server 2003, Visual studio .NET). Proto je jednoduchá instalace projektů tvořených v ASP .NET. Stačí jen upravit konfigurační soubor XML pro cílový server a aplikaci zkopírovat. **[3]**

# 4 Java Server Pages

## 4.1 Rozdělení Javy

Sun Microsystems již před nějakou dobou rozdělil Javu jako platformu do několika částí, které mají specifické použití:

- Java Micro Edition (J2ME) – určená pro drobná a mobilní zařízení (PDA, mobily)
- Java Standard Edition (J2SE) – základní knihovny, využívané především pro „desktopové“ aplikace
- Java Enterprise Edition (J2EE) – určená pro serverové aplikace

## 4.2 Java enterprise edition (J2EE)

Spíše než o platformu jde o specifikaci, jak vytvářet rozsáhlé vícevrstvé aplikace. Umožňuje vystavět aplikaci pomoci komponent a služeb, které k těmto komponentám nabízí. Využívá také výhod J2SE, jako je přenositelnost mezi různými architekturami počítačů, zabezpečení aplikací a další. Důvod, proč je J2EE také velmi oblíbené je tvorba aplikací z komponent. Protože ne každý vývojář musí dělat pro funkce, které do aplikace chce implementovat vlastní komponentu, ale může využít již dříve vytvořených. J2EE také nabízí pro komponenty služby, které jsou již automaticky zajišťovány proto, aby se vývojář mohl zaměřit na uživatelský cíl aplikace a ne na tvorbu podpůrných funkcí (správa životního cyklu nebo správa transakcí). Nyní nastíním co je to komponenta a co kontejner.

### 4.2.1 Komponenta

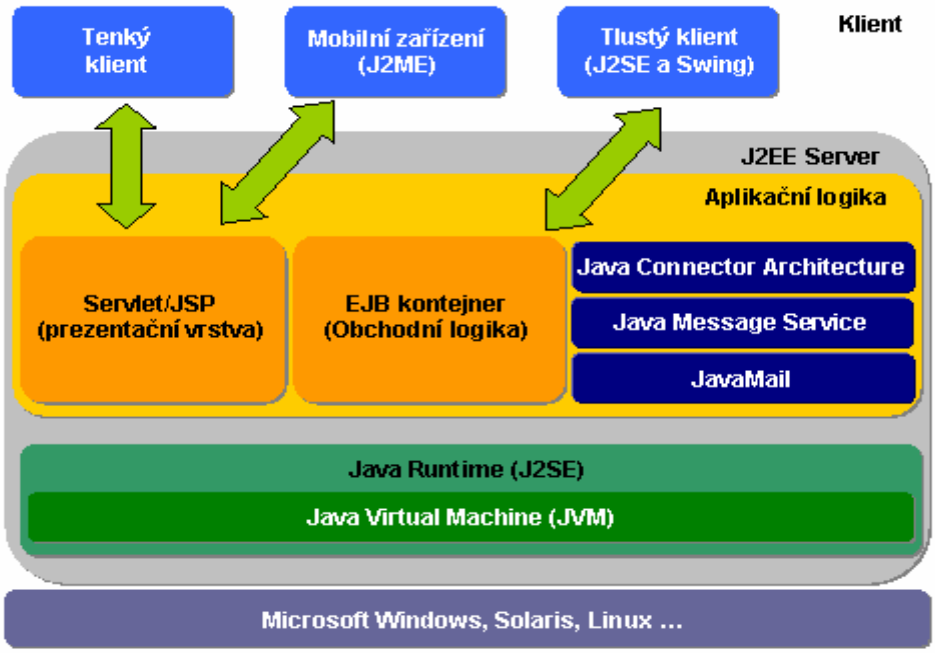
Komponenta jako taková je vlastně každá Javovská třída, ale v enterprise prostředí byla dána jasná definice z roku 1996 od *European Workshop on Component-Oriented Programming* (ECOOP).

Softwarový komponent musí mít jasně definované rozhraní a explicitně vyjádřené závislosti. Musí být možné ho nasadit nezávisle jako produkt třetí strany.

Komponent má obvykle několik metod, určité vlastnosti a může generovat události. Komponent může mít také logické propojení na jiný komponent, tato závislost musí být explicitně dokumentována.

### 4.2.2 Kontejner

Kontejner je část serveru, která poskytuje úplné aplikační prostředí, v němž řídí životní cyklus nasazených entit a poskytuje jim různé služby. J2EE obsahuje několik kontejnerů včetně JSP kontejneru, servlet kontejneru (dohromady tvoří webový kontejner) a Enterprise JavaBeans kontejneru (obsluhující komponenty EJB). Takový kontejner může být například Apache Tomcat.




**Obrázek 4-1:** Platforma J2EE skládající se z aplikačních komponent a kontejnerů.

Následuje popis komponent a služeb kontejneru (viz. Obrázek 1):

- Servlet – Java třídy generující dynamický obsah stránek
- Java Server Pages – HTML stránky používající jazyk Java pro generování dynamického obsahu
- Enterprise JavaBeans – definuje komponenty na straně serveru a jejich spolupráci s aplikačním serverem
- Java Messaging Service (JMS) – asynchronní zasílání zpráv
- Java Connector Architecture - Přístup k jiným aplikačním systémům

Další služby kontejneru mohou být následující:

- Java Authentication and Authorization Service (JAAS) – zajištění bezpečnosti
- Java Transaction API (JTA) – transakční služba
- Java API For XML Processing (JAXP) – podpora práce s XML
- A další **[4]**

# 4.3 Java Server Pages (JSP)

JSP jsou HTML stránky, do kterých se pomoci speciálních značek vkládá kód Javy. Tento kód potom tvoří dynamický obsah webových stránek. Je více možností jak tvořit JSP stránky, například použití standardních programů pro tvorbu webových stránek (homesite, dreamweaver, macromedia flash aj.) a dopsání kódu Javy nebo přímo psát stránky třeba pomoci IDE pro vývoj webových aplikací (Elipse, JDeveloper aj.). Použití JSP stránek je vhodné pokud převládá statický obsah stránek v opačném případě je lepší používat servlety.

## 4.3.1 Princip činnosti

Uživatel požádá o webovou stránku, která je tvořená jako JSP (tzn. Stránka s příponou .jsp), klient vytvoří požadavek (request) a pošle ho serveru, ten zjistí, že byl vyslán požadavek na JSP stránku a přesměruje JSP soubor do servlet stroje. Pokud je soubor volaný poprvé, pak stroj soubor překontroluje a generuje speciální servlet, kde všechno HTML je uloženo v příkazech out.println().

Zdrojový kód servletu je zkompilován a je vytvořen soubor s příponou .class, spustí se instance servletu. Výsledkem servletu je prostý HTML kód poslaný uživateli zpět do prohlížeče.

### 4.3.2 Skriptovací značky

V JSP stránkách se používají tři druhy skriptovací značek. Prvním z nich jsou výrazy, které se vkládají do výstupu generované stránky, jejich syntaxe je takováto:

<%= výraz v jazyce Java %>

Tento výraz se vyhodnotí, převede se na řetězec a vloží se do stránky. Dalším typem jsou skriptlety, které umožňují vkládat složitější části kódu v jazyce Java (cykly, podmínky aj.), jejich syntaxe je takováto:

<% kód v jazyce Java %>

A poslední značky slouží pro globální deklarace entit, je – li servlet tvořen tak, aby pro každý požadavek bylo vytvořeno nové vlákno, ale instance servletu je pouze jedna, pak jsou tyto deklarované entity dostupné ve všech požadavcích. Syntaxe je:

<%! deklarace entity nebo entit %>

### 4.3.3 Direktivy

Slouží k ovlivnění celkové struktury servletu a jeho funkčnosti. Zápis je tento

<%@ direktiva attribute="hodnota" %>

Jako direktiva muže být použita jedna ze tří:

- page – nastavuje, které třídy servlet rozšiřuje nebo které knihovny importuje
- include – vkládá soubory do stránek JSP v době kdy je stránka překládána do servletu, může obsahovat i JSP konstrukce

- taglib – používá se při rozšíření množiny značek o vlastní uživatelem definované značky. Tyto vlastní definované značky se ukládají v tzv. popisovači (Tag library descriptor, přípona .tld).

### 4.3.4 Akce

Jsou to buď uživatelem definované nebo standardní (s předponou jsp) komponenty. Lze si pomoci nich vkládat do jedné stránky výstup z jiné nebo využívat navržených tříd JavaBeans. U vlastních akcí se předpona určuje direktivou taglib a atributem prefix.

### 4.3.5 Komentáře

Jsou dva druhy komentářů jeden skrytý komentář, sloužící ke komentování samotných JSP stránek (tento se nevyskytuje ve výstupním HTML) a dále výstupní komentář, to je standardní komentář HTML, který se zahrne i do výsledné HTML stránky.

### 4.3.6 Implicitní objekty

Slouží pro zpřístupnění některých služeb aplikačního serveru JSP stránkám, jsou to objekty, které vývojář nemusí nijak inicializovat, nebo se o ně starat, protože jejich existenci musí zajistit daný aplikační server. Jsou to například

- request – reprezentuje analyzovaný požadavek
- response – reprezentuje tvořenou odezvu
- session – slouží pro sledování relace (sezení) klienta
- application – objekt, který je sdílený pro všechny požadavky na danou aplikaci (můžeme zde ukládat data a dále je získávat v ostatních stránkách) **[5]**

# 5 Rozdíl ASP .NET a JSP J2EE

Po pročtení nemalého množství materiálů porovnávajících frameworky .NET a J2EE jsem zjistil, že porovnání samotných technologií tvorby webových aplikací jakými jsou ASP .NET a JSP J2EE je docela obtížné. Proto se pokusím z dosažených výsledků nastínit

| **Active Server Pages .NET** | **Java Server Pages J2EE** |
|---|---|
| Může využívat celou řadu programovacích jazyků, ale nejpoužívanější jsou C# a VB .NET. | Java |
| Visual Studio .NET je plně integrované se serverovými systémy. Hodně vývojových prostředí se inspiruje v tomto studiu. | Není jednotné vývojové prostředí, liší se specifickými výrobci serverů, kteří nabízejí integrovaná prostředí pro vývoj. (Eclipse aj.) |
| Kód se kompiluje do Microsoft Intermediate Language (MSIL). | Java bytecode |
| Zdrojové soubory aplikace jsou v Assemblies a metadatech. | JAR, WAR, EAR. |
| Relativně jednoduchá konfigurace aplikace běžící na serveru. | Složitá konfigurace odvíjející se od používaného serveru. |
| Web Forms, pracují na základě Windows Forms. Nabízí hodně rozšířených funkcí na základě nového modelu GDI+. | Obdobné funkce nabízí JavaServer faces (ale ještě není zažité vývojové prostředí) nebo struts. |
| Silná integrace s XML. XML je integrováno do GUI, práce s datovými zdroji, nternetovými protokoly, vnitřní komunikace v aplikaci atd. | XML podporováno spíše na základě pársování. |

| | |
|---|---|
| Přímá podpora Web services – technologie, která umožňuje integrovat libovolné aplikace provozované na různých platformách a ovládat je prostřednictvím webového rozhraní. | Práce s web services není v J2EE standardizována. |
| I když v popisu .NET frameworku je přenositelnost na různé architektury a OS, ASP funguje pouze na windowsech a Linuxech v rámci mono projektu. | Přenositelnost na různé operační systémy, kde je Java virtual machina (JVM). |
| Špičkový vývojový tým. | Většinou open – source vývojáři |
| Zpoplatněné prostředky pro vývoj, správu a provozování aplikací. I když se objevily verze, které lze používat bez poplatků. | Většinou bezplatné prostředky vyvíjené jako open – source (Elipse, Apache Tomcat) |
| V neposlední řadě je tato technologie relativně mladá, a poučila se ze svých chyb i z chyb konkurenčním technologií. | Tato technologie je o několik let starší než ASP .NET, proto vychází v celkovém porovnání hůře. |

Ve srovnání vychází o něco lépe ASP .NET, je to také tím, že je to docela mladá technologie, vhodná pro vývoj aplikací pro komerční účely. A to nejen díky kvalitní technologii, ale také velmi dobrého zázemí (MSDN). Samozřejmě i velké a obsáhle aplikace se tvoří v Javě, ale ASP .NET je rychlejší jak pro tvorbu, tak pro konfiguraci na serveru a i rychlost samotné aplikace (záleží také hodně na vývojáři).

# 6 Framework Struts

V této části dokumentace bych chtěl ve stručnosti nastínit co je to framework Struts, protože je to stavební kámen mé aplikace. Struts je produktem nadace Apache Software Foundation (tvůrci kontejneru Apache Tomcat), který vznikl před několika lety. Je to webové rozhraní pro tvorbu webových aplikací podobné .NET webForms. Umožňuje snadnou vícejazyčnou podporu pomocí souborů, které nahrazují určitou formulaci v JSP příslušným řetězcem v těchto souborech (typicky každý soubor pro jeden jazyk), s příponou .properties. Další výhodou je zpřehlednění kódu JSP stránek, místo typického HTML se využívají speciální tagy a struts formuláře, ke kterým se dostanu později.

## 6.1 Technika MVC

Jedná se o design pattern, neboli jeden z popisů doporučených programátorských postupů a technik. Této techniky využívá i Struts. Pak aplikace napsaná ve Strutsu se skládá ze tří částí:

- model – komponenty v JavaBeans, typicky jsou to třídy v Javě, které mají metody get() a set(), které se dále využívají v controlu
- view – generování výsledné podoby výstupu, v případě webové aplikace HTML. Využívají se JSP stránky se speciálními Struts značkami, které jsou uloženy v popisovačích (TLD)
- controller – o kontrolu a řízení celé aplikace se starají tzv. akční třídy, jsou to akce psané v Javě, které načítají data z databáze, kontrolují správnost přihlašovacích údajů a předávají řízení na základě informací přicházejících s HTTP požadavkem.

## 6.2 Princip činnosti

### 6.2.1 Web.xml

Základem celé aplikace je třída ActionServlet, patřící do vrstvy controller, stará se o vytváření instancí akčních tříd a rozdělování práce těmto akčním třídám. V základní implementaci knihovny Struts je již dostačující verze ActionServletu, ale samozřejmě se dá vytvořit potomek s jinými vlastnostmi, který musí být namapován v souboru web.xml.

Základní konfigurace Struts je v souboru web.xml, kde jsou připojeny popisovače (TLD) a konfigurační soubor vlastní webové aplikace struts-config.xml. Jelikož je Struts založen na akcích, může být v tomto souboru nastaveno při jaké příponě souborů v URL volat danou akci (např. při URL www.aplikace.cz/index.do se po nastavení příslušného mapování v config-struts.xml volá akční třída indexAction.class) a která bude počáteční stránka aplikace (tradičně index.jsp). Při prvním dotazu se inicializuje ActionServlet, načte se konfigurace aplikace ze souboru struts-config.xml, všechny lokalizační soubory, plug-iny a aktivují se databázová připojení.

### 6.2.2 Struts-config.xml

Tento soubor je hlavním konfiguračním souborem aplikace. V první části se definuje připojení k databázi (cesta k databází, ovladač použitý k připojení). V další části je definice formulářů použitých v aplikaci. Tyto formuláře jsou Java třídy, potomci třídy ActionForm, které obsahují metody set() (na ukládání dat z formulářů) a get() (které jsou použité pro používání hodnot nastavených metodou set() v jiných částech aplikace). Hlavní částí je mapování akcí, tj. při jaké akci zavolané v JSP stránce se bude volat daná akční třída. Pak už následuje vložení plug-inů, které budeme chtít používat pro usnadnění práce (např. validator, který kontroluje správnost hodnot zadaných do formulářů aj.) a cesta k souboru pro vícejazyčnou podporu.

### 6.2.3 Popisovače

Nebo jak bylo už dříve zmíněno tag library deskriptor (TLD). Uvedu standardní TLD, které jsou součástí frameworku Struts, podrobnější popis bude až v další části dokumentace, týkající se implementace, a to hlavně u těch které jsem používal v aplikaci.

- bean tags – rozšíření tradičních <jsp: …> značek
- html tags – slouží k vytvoření HTML elementu na výstupu
- logic tags – používají se pro podmíněné řízení toku, iterování přes kolekce aj.
- tiles tags – slouží k přizpůsobení vzhledu danému uživateli, pomoci skládání jednotlivých částí stránek
- a jiné **[6]**

# 7 Systém elektronické aukce

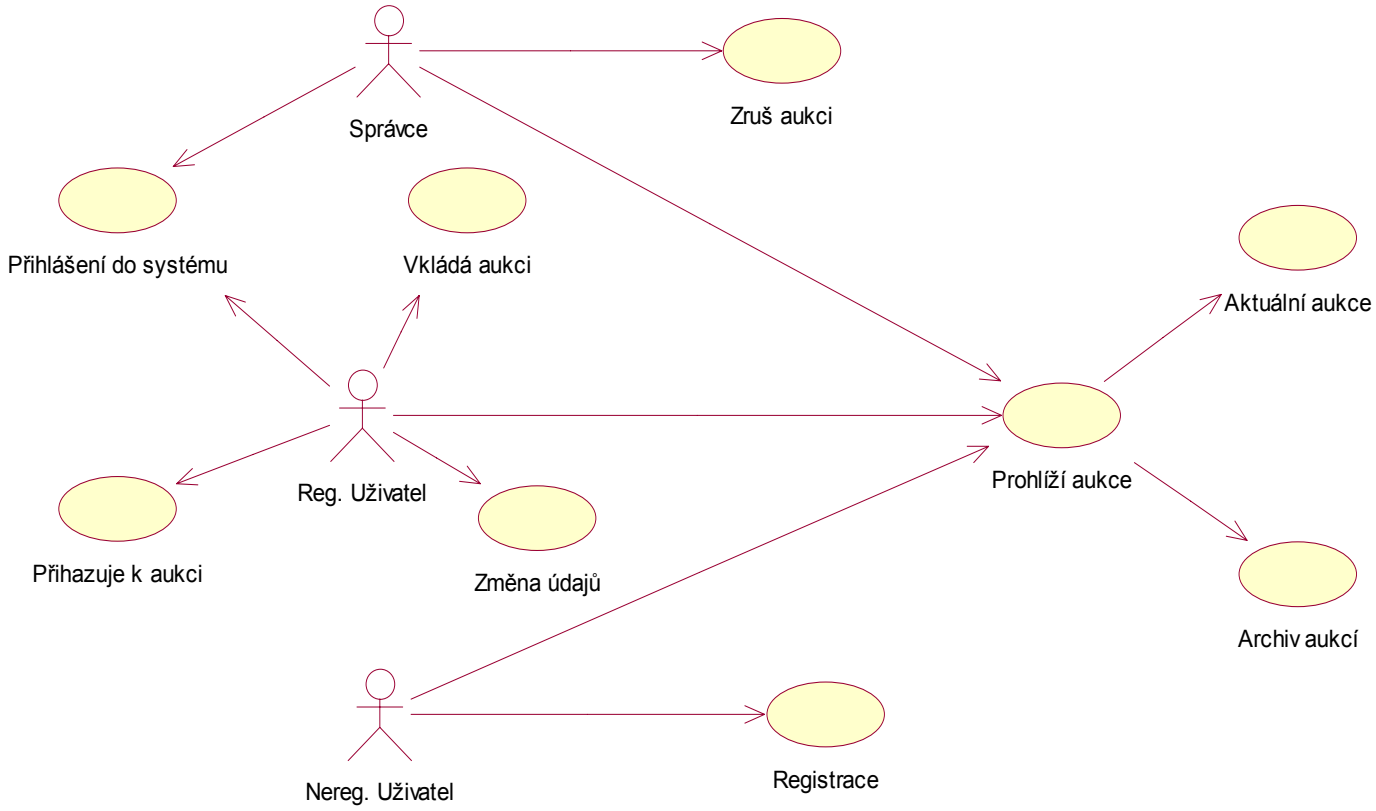
## 7.1 Popis projektu

Mým projektem je online systém elektronické aukce s využitím programovacích prostředků Struts. Je to systém, který umožňuje uživatelům s příslušnými právy sledovat a dražit v určitých aukcích. Základem pro dražbu a přidávání nových položek do aukčního systému je registrace uživatele. Po vyplnění příslušných údajů je uživatel zaveden do tabulky uživatelů v databázi. V mé aukci jsou dva druhy registrovaných uživatelů a to administrátoři a normální uživatelé. Administrátoři nemají možnost dražit, ale mají privilegium odstraňovat tzv. nedůvěryhodné aukce (aukce jejichž zadavatel nesplnil dodací podmínky nebo se k zadávaným aukcím nevyjadřuje). Druhý registrovaný má možnost aukce jak prohlížet, tak i přidávat a k již běžícím aukcím přihazovat. Neregistrovaný uživatel může pouze aukce prohlížet. Při vkládání uživatele do databáze je každému přiřazeno identifikační číslo, které slouží k lepšímu sledování uživatele v aplikaci.

Každý přihlášený uživatel může sledovat své probíhající aukce (tzn. Aukce, které buď sám zadal, nebo u kterých přihazoval), pokud je již daná aukce ukončena, tak při prvním přihlášení uživatele po skončení aukce, uživatel obdrží zprávu, že jeho aukce byla ukončena. Pokud aukci zadával, pak bude na email, který zadal při registraci, odeslán kontaktní email na výherce aukce. V opačném případě obdrží zprávu o přeposlání kontaktního emailu zadavateli.

Pokud uživatel přidává aukci do systému je dotázán na délku trvání dané aukce (tj. jak dlouho se bude moci k aukci přihazovat), konečný čas aukce se bere od času zadání aukce plus příslušného časového úseku. Při vkládání aukce do databáze se aukcím přidělí jednoznačné číslo aukce. Aby bylo možno aukci sledovat, je u každé aukce identifikační číslo zadavatele (identifikační číslo uživatele, které mu je přiděleno při registraci) a identifikační číslo kupujícího (pokud existuje). Aukce obsahuje příznak platnosti, který se mění až po zobrazení aukce zadavateli při přihlášení.

Přihazovat mohou pouze uživatelé, kteří jsou v systému zaregistrování, na to je systém upozorní. Je – li uživatel přihlášen zvolí částku, kterou přihodit a tím se do tabulky aukcí přidá také uživatelské číslo kupujícího, které se mění podle toho, který uživatel je aktuálním kupujícím.




**Obrázek 7-1: Use case diagram**

## 7.2 Použitý kontejner

Pro kontrolu aplikace jsem používal aplikační server Apache Tomcat od nadace Apache Software foundation, sloužící pro běh JSP a Java Servlets. Nejdříve bylo potřeba Tomcat konfigurovat, tzn. nastavit cesty ke standardním knihovnám Javy J2SE. Poté jsem musel nastavit uživatele a roli uživatele v souboru tomcat-users.xml, tak abych mohl server používat. Server využívá port 8080, a k aplikacím běžícím na Apache Tomcat se přistupuje přes URL *http://localhost:8080/Aplikace*.

## 7.3 Vývojové prostředí

Pro vývoj aplikace jsem použil grafické vývojové prostředí Eclipse od firmy IBM. Jelikož jsem projekt dělal s použitím frameworku Struts, musel jsem do Eclipsu nainstalovat plug-in Exadel Studio podporující Struts. Na počátku tvorby se mi zdál Exadel docela nepřehledný a samotná konfigurace aplikace pod tímto prostředím byla relativně obtížná, ale poté co jsem aplikaci dobře nakonfiguroval, byl Exadel přínosem. Exadel sám obsahoval velké množství prvků, které usnadnily práci při tvorbě aplikace. Ať už se jedná o nástroj ANT, který se používá pro kompilování zdrojových kódu, sestavení aplikace atd. nebo možnost připojit aplikaci na některý ze serverů. V mém případě Apache Tomcat, takže při změně konfiguračního souboru (struts-config.xml) se restartoval pouze Tomcat používaný Exadelem, to ušetřilo až 15 sekund při každé kompilaci.

## 7.4 Databáze

Při tvorbě databáze jsem použil databázový server MySQL. Pro přístup aplikace k databázi jsem musel:

- V konfiguračním souboru aplikace (struts-config.xml) se nastaví adresa databázového serveru a název databáze (v mém případě *jdbc:mysql://localhost:3306/aukce*), dále uživatel a jeho heslo potřebné k přístupu k datům z databáze a driver pro práci s databází.
- Nahrát driver pro práci s databází (soubor mysql-connector-java.jar) do adresáře common kontejneru Apache Tomcat.

Potom jsem si vytvořil databázi, pomoci které jsem odzkoušel funkčnost aplikace.

# 7.5 Implementace

## 7.5.1 Formulář

I když jsem již nastínil jak funguje Struts, uvedl bych praktický příklad. Hlavní myšlenkou Strutsu je vylepšená obsluha formulářů. Pro každý formulář se musí vytvořit příslušná třída, potomek třídy ActionForm, např. RegistrationForm. Pro každou položku formuláře obsahuje metody getPolozka() a setPolozka(). Poté si v JSP stránce vytvoříme formulář s názvem registerForm (musíme mu přiřadit nějakou JavaBeans komponentu, v tomto případě RegistrationForm) a nastavíme tomuto formuláři akci, v mém případě *register* , která se bude volat po stisknutí potvrzovacího tlačítka formuláře. Pokud chci načíst z formuláře jméno, musí mít v tagu *html:text* atribut *name* hodnotu nějaké již vytvořené JavaBeans, v tomto případě registerForm a atribut *property* hodnotu položky pro kterou je definována metoda getPolozka(). Pro lepší přehlednost mapováni akce umístím obrázek, část souboru struts-config.xml, a jednotlivé atributy vysvětlím.

```
<action path="/register" name="registerForm" scope="request"
          type="cz.actions.UserRegistrationAction">
 <forward name="success" path="/pages/Vystup.jsp"/>
 <forward name="failed" path="/pages/UserLogForm.jsp"/>
</action>
```

**Obrázek 7-2: Mapování akce ze souboru struts-config.xml**

- path – akce, která se volá pro daný formulář
- name – jméno formuláře, které musí být spojeno s nějakým ActionForm, který má definované metody set() a get()
- scope – platnost dotazu
- type – jaká akční třída se bude pro tento dotaz volat
- forward – je přesměrování na stránku definovanou atributem path při splnění daného řetězce (name), který vrací akční třída

### 7.5.2 Akční třída

Akční třída je potomek Action knihovny struts. Ve verzi Struts, kterou používám (1.2) je potřeba překrýt metodu execute() typu ActionForward, která vrací forward, podle něhož se pak předává řízení dále. V akční třídě UserRegistrationAction se připojuji k databázi, zjišťuji jestli uživatel se stejným nickem v databázi již existuje, pokud ano, vrací metoda execute výsledek *failed* a uživatel se vrátí na stránku s registračním formulářem. Je – li všechno správně pak metoda vrací *success,* uživatel se přidá do databáze a je přesměrován na stránku s potvrzením registrace.

### 7.5.3 Logic

Další zajímavou částí je načítání aukcí z databáze a následné zobrazování. Struts framework konkrétně pak popisovač *logic* obsahuje tag *logic:iterate* pro načítání dat z kolekcí. Takže v akční třídě si data z databáze ukládám do dvourozměrného pole typu ArrayList a výsledný objekt si uložím do session metodou setAttribute. Při použití tagu logic:iterate musí atribut *name* odpovídat již existujícímu objektu v daném rozsahu (*scope* – session, request, page) a přes atribut *id* se dá přistupovat k dané položce v arraylistu. Iterate už pak projíždí všechny položky v daném Arraylistu a každá z položek se v tomto případě zobrazuje v tabulce.

Logic obsahuje celou řadu tagů, které usnadňují a zpřehledňují vývoj aplikace. Hojně jsem používal tag logic:present resp. logic:notPresent. Tyto tagy zjišťují jestli v daném rozsahu (*scope*) je obsažen objekt resp. není obsažen. Využívám toho třeba pro zjištění jestli je uživatel do systému přihlášen. Pokud ano, tak jaké jsou jeho práva a dále.

### 7.5.4 Bean

Tento popisovač se využívá především pro výpis hodnot. Nejvíce jsem využíval tag bean:message, hodnoty atributu *key* jsou nahrazeny příslušnými hodnotami ze souboru *MessageResources.properties* a *bean:write,* který se používá pro výpis hodnot daného JavaBeans.

## 7.6 Problémy při implementaci

Při implementaci se vyskytla celá řada problémů. Jeden z nejzapeklitějších byl se samotným Exadelem. Poté co jsem si importoval z nějakého adresáře projekt, tak si Exadel vytvořil zálohu v mém workspace. A čas od času se změny v zdrojových souborech neprojevily v aplikaci. Exadel si přestal ukládat změny do záloh ve workspace a ukládal je jenom do adresáře, ze kterého se projekt načítal. Když se mi toto stalo poprvé, trvalo mi vyřešení tohoto problému půl dne. K vyřešení stačilo zrušit projekt v Exadelu a importovat ho znovu.

Další byl se zobrazováním českých znaků po načtení z databáze. Dodržel jsem všechny pokyny, které jsem našel na webu k nastavení databáze v MySQL. Kódování znaků databáze, tabulek i sloupců tabulek jsem měl nastavené na *cp1250*, obdoba kódování stránek *windows-1250***.** Data která jsem posílal po nějakém dotazu zpět aplikaci byla také nastavena na *cp1250*. Nakonec byl problém vyřešen přidáním parametru *characterEncoding=cp1250* v nastavení driveru používaném pro připojení k databázi.

## 7.7 Závěr

Výsledkem tohoto ročníkového projektu je seznámení s technologií Java Server Pages a porovnání této technologie s obdobnými prostředky ASP .NET. Hlavní náplní mé práce bylo seznámení s programování JSP stránek a tvorbou webových aplikací. Jelikož jsem se setkal s tvorbou webu poprvé strávil jsem mnoho času pročítáním možností jak web vytvářet pomoci JSP stránek a Java servlets a právě při tomto studiu jsem narazil na framework Struts.

Při tvorbě aplikace jsem se seznámil jak s tvorbou webových aplikací pomoci JSP stránek s využitím frameworku Struts, tak se samotným programováním v Javě. Práce byla velmi různorodá a zajímavá, i když jsem se setkal z řadou problémů. Hlavně pro Struts není žádná česká dokumentace, ani internetový seriál v takové šíři, jakou jsem potřeboval pro danou aplikaci. Většinu informací jsem čerpal buď z anglických internetových knih **[7]**, ale nejvíce z různých celosvětových diskuzí, kde jsem naštěstí narazil na příspěvky týkající se stejných problémů, jako jsem řešil.

V počátcích práce s frameworkem Struts jsem shledával docela pracné mapování akcí. Pro jeden formulář vytvořit tři, čtyři soubory a ještě měnit dvakrát konfigurační soubor se mi zdálo nepohodlné. Postupem času jsem si zvykl a musím uznat, že je to lepší jak pro přehlednost kódu tak pro pochopení jednotlivých akcí. I samotné JSP stránky vypadají lépe než tradiční stránky s kódem Javy, díky popisovačům, které Struts obsahuje.

Aplikaci je zatím jen jednoduchá, na které jsem se seznamoval s těmito prostředky pro tvorbu, je celá řada rozšíření, která by se dala implementovat. Ať už pooling připojení k databázi nebo lépe řešená autentifikace a samozřejmě i rozsah celé aplikace.

# Literatura


[1] Petržalka Jiří: Dynamické webové stránky, 1.října 2004. Dokument dostupný na URL http://www.pcsvet.cz/art/article.php?id=2434 (květen 2006).

[2] Šejda Jan: J2EE, .NET a vývoj rozsáhlých systémů 2, 11.února 2003. Dokument dostupný na URL http://interval.cz/clanky/j2ee-net-a-vyvoj-rozsahlych-systemu-2 (květen 2006).

[3] Hupka Radek: Měníme stránky ASP na stránky ASP .NET, 11. srpna 2005. Dokument dostupný na URL http://www.zive.cz/h/Programovani/AR.asp?ARI=125126 (květen 2006).

[4] Šejda Jan: J2EE, .NET a vývoj rozsáhlých systémů 1, 10.února 2003. Dokument dostupný na URL http://interval.cz/clanky/j2ee-net-a-vyvoj-rozsahlych-systemu-1 (květen 2006).

[5] Bc. Maixner David: Tvorba aplikací v J2EE [diplomová práce], 2004. Masarykova univerzita, fakulta informatiky v Brně. Dokument dostupný na URL www.slunecnice.cz (květen 2006).

[6] Pešek Jan: Struts, 2004. Dokument dostupný na URL http://www.kiv.zcu.cz/~brada/vyuka/files/pia/ppp/struts/index.php (květen 2006).

[7] O'Reilly Media, Inc.: Programming Jakarta Struts, 2nd Edition, červen 2004.