TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: Informační technologie

Databáze norem

Database of Standard

Diplomová Práce

Autor: **Bc. Michal Kosek**
Vedoucí práce: Ing. Alena Gregová
Konzultant: doc. Ing. David Vališ Ph.D.

V Liberci 20. 5. 2011

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum ……

Podpis …….

Poděkování

Na tomto místě bych rád poděkoval své vedoucí diplomové práce Ing. Aleně Gregové za trpělivost, cenné připomínky a veškerou pomoc. Dále bych rád poděkoval všem, kteří se svými radami a připomínkami podíleli na zlepšení kvality této práce. Poděkování patří také všem mým blízkým za jejich podporu.

Abstrakt

Diplomová práce se zabývá možností vytvoření databáze norem, a její klientské on-line aplikace pro podporů překladů a tvorby správné a jednoznačné terminologie.

V první části jsou popsány normy a jejich vlastnosti. Dále následuje výklad problematiky jejich překladů, včetně tvorby terminologie. Je vysvětlen problém nežádoucích synonym v terminologii, a potřeba jejich jednoznačnosti u všech použitých termínů. Je zde vyslovena potřeba vytvoření slovníku termínů a jejich originálních ekvivalentů.

Druhá část popisuje výběr programovacího jazyka, prostředí internetu a tvorbu webových stránek včetně vlastností používaných protokolů.

Dále následuje detailní rozbor implementace aplikace pomocí zvolené technologie pro realizaci stránek a vybrané vlastnosti zvoleného aplikačního serveru.

Další část obsahuje návrh databáze a popisuje některá specifická řešení zvolená pro realizaci, především způsob zabezpečení údajů uložených v databázi a ochranu proti spamovacím robotům.

Na konci diplomové práce je vysvětleno, jak tuto aplikaci správně nainstalovat na server, a jak ji správně použít z hlediska uživatele.

Klíčová slova:

Normy, Databáze, MySQL, Java, JSP, Apache Tomcat

Abstract

The thesis is about possibility of creating a database of standards, and its client's on-line application for support the translation of the correct and unambiguous terminology.

The first part is described the standards and their properties. Interpretation of the translation issues including the creation of terminology are in next. It is explained the problem of unwanted synonyms in terminology and the need for clarity on all the terms which are used. It is necessary to create a dictionary of terms and their genuine equivalents.

The second part is described the choice of programming language, the Internet and creating Web pages, including properties of used protocols. Detailed analysis of the implementation of applications used technology for implementation of selected sites and selected properties of the application server are stated next.

Design of database is in next part of search and it is described some specific solving for the implementation, the way of security of data stored in the database and protect against spam robots.

How install the application to correctly on the server, and how it used to correctly by user's perspective is explained on the end of this thesis.

Keywords:

Standard, Database, MySQL, Java, JSP, Apache Tomcat

Obsah

Seznam Obrázků

Seznam Tabulek

Úvod

V současné době se v České republice vydávají české státní normy (ČSN), na tuto činnost dohlíží Úřad pro technickou normalizaci, metrologii a státní zkušebnictví. Norma vzniká buď jejím vytvořením anebo překladem či převzetím. České státní normy vytvářejí jen okolo 10 % norem, především v oblastech, kde žádná norma na daný účel neexistuje. Situace tedy nastává poměrně zřídka. Většina norem ČSN vzniká překladem či převzetím z jiných jazyků, kde se přebírají z jiných ať už národních, mezinárodních či globálních institucí zabývajících se tvorbou technických norem. Převzetí znamená, že norma dostane české označení a její úvodní list je v češtině, zbytek je v originálním jazyce (většinou angličtina nebo francouzština). Překlad normy do češtiny je časově náročnější a má svá úskalí. I přes to je formou překladu převzato asi 60% zahraničních norem. Diplomová práce popisuje problematiku tvorby správné odborné terminologie k překládaným normám.

Cílem této diplomové práce je seznámit čtenáře s problematikou norem a jejich překladů a posléze popsat návrh a implementaci aplikace, která by kromě evidence norem samotných, umožňovala vytvářet terminologický slovník, do kterého je možné vkládat a poté prohlížet existující odbornou terminologii, která se v normách používá. Mezinárodní normalizační instituce takovýmito slovníky disponují.

1 Technické normy

1.1 Co jsou to technické normy

Technická norma je podle ČSN EN 45020 „dokument vytvořený na základně konsenzu a schválený uznaným orgánem, poskytující pro všeobecné a opakované použití pravidla, směrnice nebo znaky pro činnosti nebo jejich výsledky a zaměřený na dosažení optimálního stupně v dané souvislosti“ [20].

Normy slouží k usnadnění volného pohybu výrobků a služeb v mezinárodním obchodu, usilují o to, aby výroba byla racionální, aby se ochrana životního prostředí a konkurenceschopnost vzájemně podporovaly a také, aby na vnitřním trhu byli spotřebitelé dostatečně ochráněni.

V nynější době lze na normy pohlížet jako na kvalifikované doporučení, nejsou pro nikoho samy od sebe závazné. Rozhodnutí, zda se bude nějaký subjekt řídit podle norem, je jeho dobrovolné rozhodnutí, které je ale často v jeho vlastním zájmu.

Normy jsou dokumenty, které jsou volně dostupné veřejnosti. Z toho vyplívá, že jejich znění ve všech fázích jejich existence (vznik a následné používání) je dostupné.

Jsou to dokumenty, které jsou založený na tom, že všechny strany, které se podílely na jejich vzniku, souhlasily s jejich zněním a všemi zásadními otázkami jejich řešení. Tímto se normy odlišují od právních předpisů, které často vznikají bez souhlasu či projednání všech subjektů, na které má jejich znění nějaký dopad.

Existují různé druhy norem, které se odlišují podle toho, pro jaký účel využití jsou určené (bezpečnostní předpisy, normy postupů/služeb, řízení jakosti, rozhraní, zaměnitelnost, terminologické, základní, zkušební, normy výrobků).

Přínosy a vlastnosti technických norem:

a) Zjednodušují a snižují rozmanitost výrobků a činností, tím se zvyšuje možná zaměnitelnost a posiluje konkurence.

b) Slouží jako úroveň, ke které se porovnávají výrobky nebo služby.

c) Stanovují kritéria bezpečnosti.

d) Podporují vztah mezi náklady a jakostí.

e) Mohou se stát závaznými v obchodních smlouvách mezi dodavatelem a odběratelem.

f) Často povinně vyžadovány u veřejných zakázek.

1.1.2 Kde získat technické normy

Jak bylo zmíněno výše, normy jsou veřejně přístupné, ovšem nejsou volně přístupné. Organizace, které je vydávají, musejí svojí činnost z něčeho hradit. Normy jsou tudíž dostupné za poplatek.

Např. ČSN, která vydávání nových norem oznamuje na svém věstníku, je zprostředkovává ve formě elektronické, tedy prostřednictvím služby ČSN online a dále v tištěné podobě, kde tisk a následnou distribuci vykonávají smluvní partneři ÚNMZ v jednotlivých krajích České republiky [19].

1.1.3 Závaznost technických norem

Obecně v České republice stejně tak jako všude jinde ve světě normy závazné samy od sebe nejsou. Novela zákona č.22/1997 Sb. (provedená zákonem č.71/2000 Sb.) výslovně uvádí, že česká technická norma není obecně závazná. Z toho vyplývá, že ČSN nejsou považovány za právní předpisy a není stanovena povinnost jejich dodržování. To ovšem neznamená, že by povinnost řídit se normami nemohla vzniknout.

Podobně tak jako v jiných zemích existují případy, že povinnost dodržovat požadavky a poznatky uvedené v normách vyplývá z jiného právního aktu, jako je:

a) smlouva mezi účastníky obchodního vztahu

b) pokyn nadřízeného - v zaměstnaneckých vztazích může vzniknout povinnost řídit se ustanovením norem, jestliže zaměstnavatel s těmito normami zaměstnance řádně seznámí.

c) rozhodnutí správního orgánu - z rozhodnutí správního orgánu na základě zmocnění uvedeného v zákoně. Případy časté ve stavebnictví.

d) právní předpis - např. Český právní řád má v sobě obsaženu řadu předpisů, které stanovují přímo či nepřímo povinnosti řídit se nějakou technickou normou.

1.2 Jak vznikají normy

V úvodu je vhodné konstatovat, že v dnešní době tvorba původních ČSN (tedy norem, které nejsou přejaté) tvoří pouze asi 10% ze všech norem, které ČSN za rok vyprodukuje. Naprostá

většina z více jak 2000 norem, které jsou každý rok vydávány, jsou normy přejaté od evropských a mezinárodních organizací, na jejichž tvorbě se prostřednictvím ÚNMZ více či méně podílejí i odborníci z ČR.

Zjednodušeně lze postup tvorby každé technické normy rozdělit na několik následujících kroků [14].

a) Návrh na tvorbu normy - Podmět k vypracování technické normy může podat každý. Prostřednictvím ÚNMZ je možné navrhnout zpracování mezinárodní nebo evropské normy.

b) Posouzení návrhu - v České Republice návrh posuzuje příslušná národní Technická normalizační komise.

c) Zpracování návrhu normy - ÚNMZ sám nezpracovává návrhy ČSN, jejich zpracování si organizuje a zajišťuje smluvně. Součástí tohoto smluvního ujednání bývá dohodnutý zpracovatel, termíny jednotlivých etapy zařazeného normalizačního úkolu a také způsob financování. Údaje o zahájení a plánovaném postupu prací na nové nebo revidované normě zveřejňuje ÚNMZ na svých webových stránkách.

d) Dohodnutý zpracovatel vypracuje vlastní návrh původní ČSN - první návrh evropské nebo mezinárodní normy je tvořen v rámci pracovní skupiny.

e) Připomínkování návrhu normy - postupné návrhy původních ČSN i návrhy evropských a mezinárodních norem se projednávají v Technických normalizačních komisích s cílem dosáhnout shody o užitečnosti navrhovaného řešení pro všechny zúčastněné.

f) Hlasování o návrhu normy, schválení návrhu normy - návrh evropské normy se schvaluje v evropských organizacích váženým hlasováním, které v podstatě vyjadřuje hospodářskou významnost členských zemí. ČR má v tomto systému 12 hlasů stejně jako Belgie, Maďarsko, Portugalsko a Řecko. Po schválení jsou členské země povinny je do 6 měsíců zavést do svých národních norem. V organizacích ISO a IEC je ke schválení potřeba 75% kladných stanovisek z všech hlasujících členů.

1.2.1 Proces převzetí evropské nebo mezinárodní normy, schválení původní ČSN

Povinností ÚNMZ, jako řádného člena evropských normalizačních komisí, je zabezpečit zavedení všech evropských norem do soustavy ČSN a zrušení těch národních norem, které jsou s

evropskými v rozporu. To se děje rozličným způsobem, především v závislosti na charakteru problematiky a okruhu potenciálních zájemců, resp. uživatelů. V každém případě se evropské normě udělí status české národní normy, a to jedním z následujících způsobů[21].

a) převzetím překladem, tj. vydáním ČSN, obsahující národní titulní stranu, národní předmluvu, úplný překlad originálu přejímané normy a národní přílohu (je-li potřebná)

b) převzetím originálu, tj. vydáním ČSN obsahující národní titulní stranu, národní předmluvu, přetisk anglické, popř. anglické a francouzské verze přejímané normy a národní přílohu (je-li potřebná)

c) převzetím schválením k přímému používání oznámením ve Věstníku, tj. "vydáním" obálky s českým názvem a označením převzaté normy, do které je vložen anglický originál přejímané normy.

Projednaný konečný návrh ČSN, a to jak původní, tak i převzaté evropské nebo mezinárodní normy předá zpracovatel ke schválení ÚNMZ. Součástí ratifikace normy je ověření naplnění zadání úkolu, metodická kontrola, a také návrhy zrušení překonaných a konfliktních již existujících norem.

Plnění programu tvorby norem se po stránce odborné spoléhá na Technické normalizační komise, které působí jako poradní orgán ÚNMZ. Přístup k nim je zcela otevřený a každý, kdo má zájem v nich pracovat, se může stát jejich členem. Jsou v nich zastoupeny různé zájmové skupiny- výrobci, spotřebitelé, obchodní organizace, školy, veřejná správa, výzkum atd.

Aktivní účast v těchto výborech přináší členům aktuální informace o dění v jejich zájmovém oboru. Prostřednictvím návrhů norem lze efektivně sledovat technický vývoj, a mnohé vyčíst o úmyslech příp. konkurentů na trhu.

1.3 Problematika přejímání norem a jejich překladů

Původní české technické normy je možné vytvářet pouze pro oblasti, pro které neexistují normy mezinárodní nebo evropské. Takovéto normy mají označení ČSN (např. ČSN 73 4301 - Obytné budovy). Ovšem dnes tvoří pouze cca 10% z celkové produkce technických norem [7].

Mezinárodní či evropské normy (označované jako ISO, EN, IEC, ETSI), které jsou přejaty do soustavy národních norem, kde se poté stávají i normami národními.

Označení se utvoří ze značky české technické normy a značky z přejímané normy např. ČSN EN, ČSN ISO, ČSN EN ISO, ČSN IEC, ČSN ETS.

Podíl přejímaných norem tvoří cca 90 % z celkové roční produkce českých technických norem. Současně s převzetím těchto norem do naší národní soustavy norem musí rušit normy zastaralé anebo konfliktní původní české technické normy.

Technické normy lze přejmout do soustavy ČSN několika způsoby:

a) překladem (cca 60% z celkového objemu přejatých norem) tzn., že v české normě za národní titulní stranou (stranami) s potřebnými informačními údaji v českém jazyce následuje text v českém jazyce doplněný v případě potřeby o národní přílohu.

b) převzetím originálu tzn., že v české normě za národní titulní stranou (stranami) s potřebnými informačními údaji v českém jazyce následuje text anglického (případně i francouzského) originálu doplněný v případě potřeby o národní přílohu.

c) schválením k přímému použití tzn., že přejmutí evropské normy je vyhlášeno ve Věstníku ÚNMZ a pokud odběratel normu opravdu vyžaduje, obdrží text originálního znění vložený do obálky s názvem a označením normy v češtině.

Překlad norem probíhá většinou buď z anglického, nebo francouzského originálu, především v těchto jazycích jsou publikovány normy, které jsou přejímány do ČSN.

Česká republika má povinnost (stejně jako všichni ostatní členové Evropské unie bez ohledu na velikost a vyspělost národních ekonomik) převzít zpravidla do 6 měsíců do své národní soustavy všechny normy evropské.

V současné době je soustava českých technických norem v souladu se soustavou norem evropských. To znamená, že veškeré evropské normy jsou průběžně přejímány do národní soustavy při současném zrušení těch národních norem nebo jejích částí, které jsou s přejímanými evropskými normami v rozporu. Postup pokračuje plynule tak, jak jsou zpracovávány a schvalovány normy nové.

České republika nemá povinnost převzít do své národní soustavy norem jakékoli normy mezinárodní. Rozhodnutí, zda bude nějaká norma převzata do národní soustavy norem ČSN, se řídí národními potřebami.

Každá norma pracuje s určitou terminologií, pro správný výklad normy je ale nutné, aby terminologie byla vysvětlena. Terminologie bývá zpravidla vysvětlena ve třetí kapitole normy (takto to mají všechny normy ČSN a stejnou zvyklost dodržuje i většina mezinárodních organizací zabývajících se vydáváním norem).

Část normy zabývající se výkladem terminologie obsahuje vždy jednotlivá slova a v případě normy vydané v originálním jazyce, ve kterém byla napsána a schválena obsahuje výklad těchto slov. V případě, že je norma přejata z jiného jazyka, obsahuje dále, z jakého slova byl termín přeložen.

Při překladu normy ovšem může nastat situace, kdy jazyk do kterého je určitá norma překládána nedisponuje vhodným ekvivalentem. V tom případě musí zpracovatel nějaký vhodný vymyslet. Problém spočívá v tom, aby dva různé pojmy neznamenali totéž. A to jednak v normě samotné, ale i mezi různými normami. Problém se řeší tím, že zpracovatel je odborník, který se v dané problematice orientuje. Poté normu schvaluje Technická normalizační komise složená z různých zájmových skupin spjatých s daným účelem, kterým se norma zabývá, tak jak je uvedeno výše v kapitole 1.2 o vzniku norem.

Pro snazší pochopení problému je zde uveden krátký příklad (Tab. 1). Pojmy *reliability* a *dependability* mají různý význam (nejsou synonyma), ale běžně jsou do češtiny překládány jako spolehlivost. Jak je ale napsáno výše, v normách je nepřípustné, aby se dvě různé věci nazývaly stejně.

Anglicky	**Česky**
reliability	spolehlivost
dependability	

Tab. 1: **Porovnání překladů**

Problém je zpravidla vyřešen přidáním vhodného přídavného jména k jednomu z termínů. Některé zahraniční organizace, které se zabývají vydáváním norem, mají on-line slovníky. Úřad pro technickou normalizaci, metrologii a státní zkušebnictví ovšem takovýto slovník zatím nemá.

Jednou z ambicí této diplomové práce je, aby vytvořená databáze norem uměla zaznamenávat k jednotlivým normám i jejich terminologii včetně definic a pojmů, ze kterých byla definice přeložena. Slovník definic poslouží k zpřehlednění a jednoduší orientaci

v terminologii norem a práce překladatelů norem, kteří musejí významy jednotlivých přeložených pojmů obtížně sledovat.

Například International Electrotechnical Commission (IEC) takovýmto slovníkem významů disponuje. Obsahuje cca 20 000 pojmů, které jsou Anglicky a Francouzsky vysvětleny a dále následuje překlad do cca 8 světových jazyků v závislosti na jednotlivých pojmech. Čeština mezi těmito jazyky ovšem není, viz Obr. 1.

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

ELECTROPEDIA

Home | Back | Print

Area	**Dependability and quality of service / Item related performance**
IEV number	**191-02-03**
EN	**dependability** the collective term used to describe the availability performance and its influencing factors : reliability performance, maintainability performance and maintenance support performance NOTE – Dependability is used only for general descriptions in non-quantitative terms.
FR	**sûreté de fonctionnement** ensemble des propriétés qui décrivent la disponibilité et les facteurs qui la conditionnent : fiabilité, maintenabilité et logistique de maintenance NOTE – La sûreté de fonctionnement est une notion générale sans caractère quantitatif.
AR	الاعتمادية الامان في التشغيل
DE	Zuverlässigkeit
ES	seguridad de funcionamiento
IT	fidatezza
PL	pewność działania
PT	dependabilidade
SV	tillförlitlighet

Obr. 1: **Náhled na výraz v on-line slovníku IEC**

2 Výběr technologii a popis prostředí Internetu

2.1 Java

Diplomová práce byla realizována v jazyce Java. K tomu vedlo hned několik důvodů. Jedná se o programovací jazyk, který se snadno učí a je objektově orientovaný. Jeho výhodou je také přenositelnost na různé platformy, protože jeho výstup po zkompilování není spustitelní soubor, ale soubor s příponou *.jar. Soubory je potom možné spouštět na každém počítači, kde je nainstalován Java Virtual Machine (JVM). JVM je dostupný na mnoha systémech, např. Microsoft Windows, Solaris OS, Linux a Mac OS [4].

Další výhodou jazyka Java je, že je vzhledem ke svému stáří (mládí) plně objektově orientovaný a celé jeho API je vytvořeno podle principů návrhových vzorů (Design Patterns). Navíc se jedná o jazyk, který je v dnešní době hodně používaný i v komerční sféře a proto je perspektivní.

S mnoha výhodami Javy existují i její nevýhody. Největší nevýhoda plyne z výše uvedený faktů. Java je naproti např. C nebo C++ pomalejší při spouštění aplikací, protože v jiných jazycích jsou aplikace již zkompilovány pro danou platformu a pak se již jen spouští. JVM musí nejdříve aplikaci zkompilovat a pak teprve spustit. Dá se říci, že je to vlastně „daň“ za přenositelnost. První verze Javy (uvedena v roce 1996) byly dvacetkrát až čtyřicetkrát pomalejší než stejný algoritmus naprogramovaný v C/C++. Díky postupným vylepšením mezi jednotlivými verzemi se rychlost postupně zvyšuje, rozdíl už není oproti jiným jazykům tak patrný jako dříve.

Java je oproti jiným jazykům o něco více náročná na operační paměť, protože kromě aplikace musí být zároveň spuštěno i celé prostředí. Fakt již v dnešní době není tak výrazný, jako dříve.

Navštívíte-li typický webový server, reaguje-li na vaše impulsy, nabízí vám různé možnosti, mění se podle vašich potřeb či si pamatuje vaše nastavení od poslední návštěvy – potom se jedná o server, který nabízí dynamický obsah. Právě díky dynamickému obsahu se mohou jednotlivé webové stránky měnit za chodu, podle preferencí jednotlivých uživatelů. V porovnání se starším stylem stránek, které jsou neměnné a které se dnes označují jako se

statickým obsahem. Statický obsah totiž zůstává stejný a to bez ohledu na to, jaké informace požaduje uživatel. U serverů se statickým obsahem se přistupuje ke změnám pouze v případě, že se k nim odhodlá správce serveru, který je pak musí i ručně provést.

Síť WWW byla ve své původní podobě striktně statickým prostředím. Bylo možné si přečíst články, prohlížet si obrázky a pak klepnout na nějaký hypertextový odkaz a přejít na další stránku. To bylo vše.

Dnešní webové servery mohou například obsahovat dynamické součásti jako:

Při navigaci mezi jednotlivými stránkami by si server měl umět zapamatovat kdo je uživatel a co doposud na stránkách vykonal.

Webový server by měl umět ukládat uživatelovi preference a zájmy a podle toho pak nabízet stránky na míru konkrétnímu uživateli.

JSP je jedna z několika technologií, které lze při tvorbě dynamických internetových stránek použít. Byla vybrána pro realizaci praktické části diplomové práce. Pro srovnání je uvedeno a krátce charakterizováno i několik dalších technologií použitelných pro realizaci stránek v dynamickém prostředí.

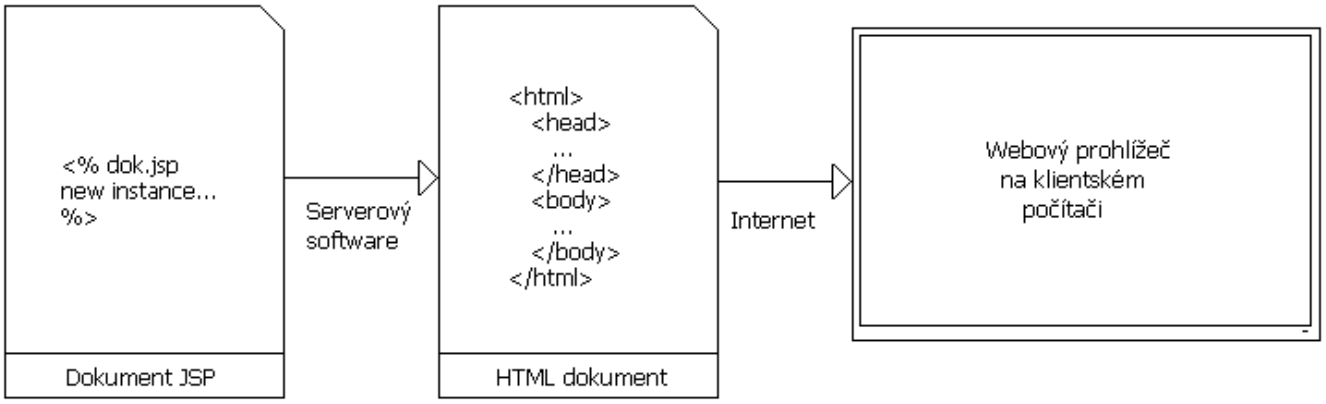
CGI - Common Gateway Interface je nejstarší technologie pro tvorbu dynamických webových stránek. Je to protokol pro napojení klienta na externí aplikaci, která na klientské požadavky sestaví stránku, kterou poté vrátí. Externí aplikace může být naprogramována v jakémkoli jazyce, typicky se používali především jazyky skriptovací, dnes je již CGI vytlačeno do ústraní.

ASP - Active Server Pages je skriptovací platforma společnosti Microsoft. Programovací jazyky, které se u ASP používají, jsou VBScript a JScript. ASP je objektově založený, není objektově orientovaný. Definuje několik základních tříd a metod, nové třídy nelze vytvářet nebo odvozovat.

PHP - Hypertext Preprocesor nejrozšířenější skriptovací jazyk pro web. Syntaxe tohoto Syntaxe jazyka je založena na několika programovacích jazycích (Perl, C, Pascal a Java). PHP je nezávislý na platformě, zpočátku vytvořen pro tvorbu osobní domácí stránky (Personal Home Page) později byl význam zkratky změněn.

2.2 O prostředí internetu

Na Obr. 2 lze vidět co se při surfování po webových stránkách děje. Uživatel sedí u počítače, který je nazýván klientský. Na tomto počítači je spuštěna aplikace, která se nazývá webový prohlížeč (Internet Explorer, Mozzila Firefox, Opera, Gogole Chrome, Safari či jiné další). Protože je program spuštěn na klientském počítači, je z hlediska vývojáře klientem. Prohlížeč pro každý příkaz uživatele (tzv. *request*) požadavek odešle prostřednictvím sítě na jiný počítač. Počítač, který požadavek přijímá, se nazývá server. Server požadavek analyzuje a odešle na něj odpověď. Odpovědí většinou bývá webová stránka. Odpověď se někdy nazývá *response*. Dokument v takovém případě není ukládán do souboru, ale je prostřednictvím prohlížeče interpretován na obrazovku.



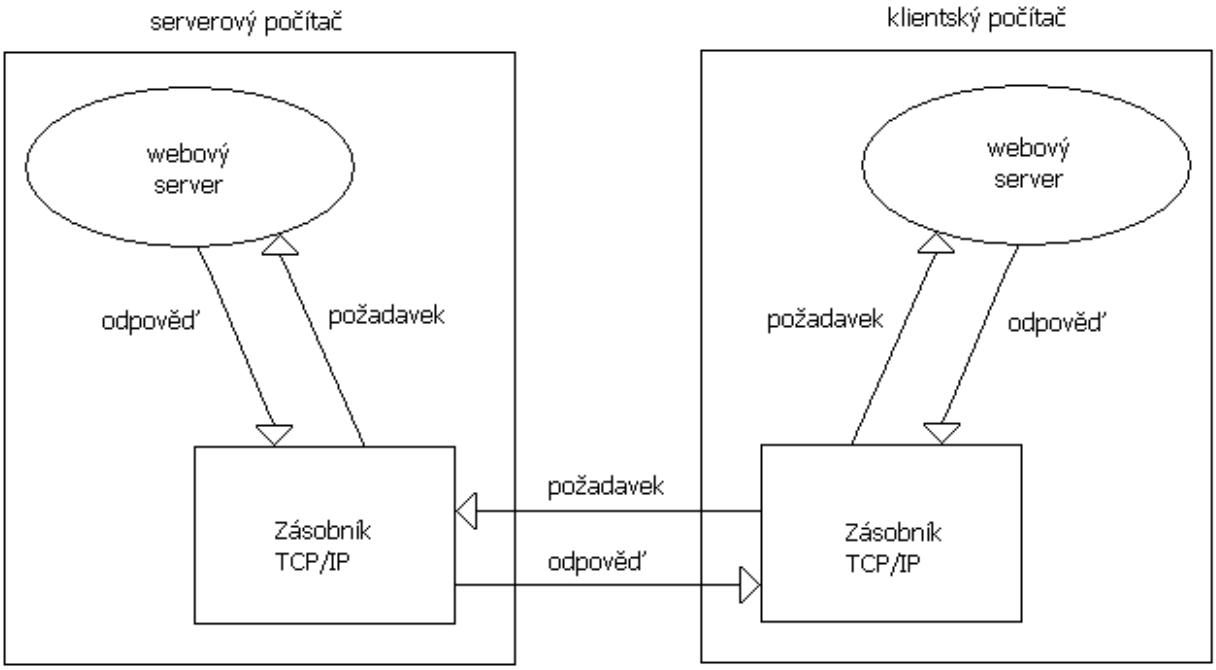
Obr. 2: **Co se děje při návštěvě dokumentu JSP**

Webovou stránkou může být jednoduchý dokument obsahující pouze text, většinou ale obsahuje dokument HTML, který kromě textu obsahuje také značky, které se nazývají *tagy*. Značky popisují vzhled a obsah stránky. Je na prohlížeči jak je interpretuje. Každý prohlížeč ovšem *tagy* interpretuje trochu jinak. Popsaný problém nastává hlavně u starších prohlížečů. V posledních letech byl problém z větší části odstraněn a nové prohlížeče již zobrazují stránky shodně.

Jazyk HTML ovšem má svoje omezení, s jeho pomocí lze vytvářet pouze stránky statické. Pro vytvoření stránek dynamických proto využijeme technologii JSP.

Server interpretuje signály získané z počítačů, u nichž sedí uživatelé procházející internetem. Rozumí jim proto, že oba počítače dodržují celosvětově platné standardy známé jako protokoly. Jedna velká sbírka protokolů, sada protokolů TCP/IP, se stará o distribuci a směrování

informací po Internetu. Název sady je zkratkou anglických názvů dvou hlavních protokolů této sady. *Transmission Control Protocol* a *Internet Protocol*. Jiný protokol, *Hypertext Transfer Protocol* (HTTP), řídí formát požadavků a odpovědí používaných v síti WWW. Všechny protokoly spolupracují v různých vrstvách. Webový prohlížeč vytvoří požadavek podle pravidel protokolu HTTP, potom požadavek předá další softwarové součásti známé jako zásobník TCP/IP (TCP/IP *stack*). Zásobník zahájí proces distribuce a směrování požadavku klienta. Poté požadavek dorazí na server, ten pomocí svého zásobníku TCP/IP opět poskládá rozkouskovaný požadavek do jednoho celku, který pak předá tomu, co je nazýváno aplikační software. Software požadavek interpretuje podle pravidel protokolu HTTP. Celý proces pro přehlednost znázorňuje Obr. 3.

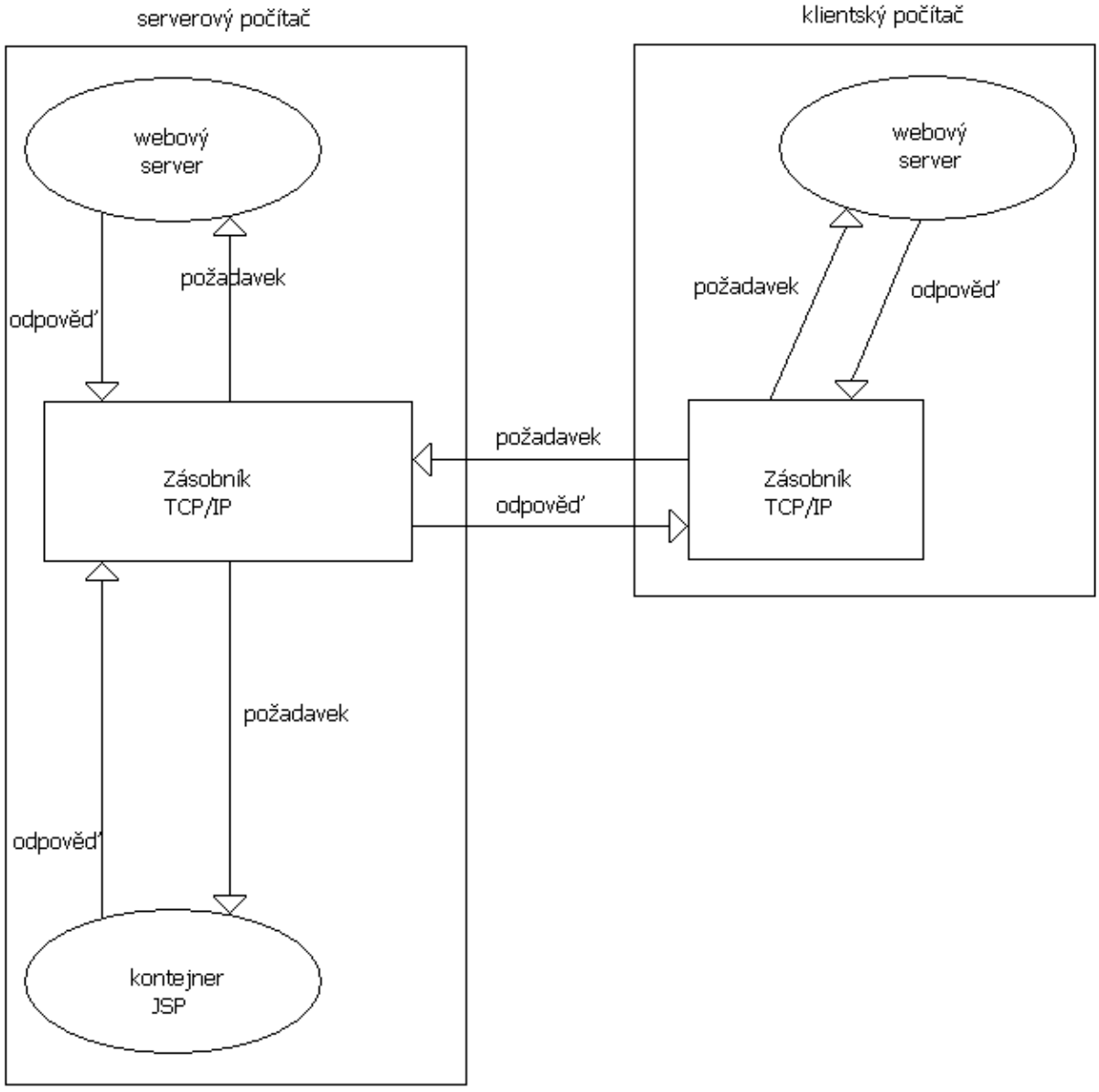


Obr. 3: **Jak klient a server používají zásobník TCP/IP**

Obrázek je trochu zjednodušující, software, kterému se obyčejně říká server, je odpovědný pouze za zpracování požadavku a odpovědi HTTP. Jestliže ale přijde požadavek na stránku JSP, předá server požadavek softwaru dalšímu, který se nazývá JSP kontejner. Ten interpretuje kód JSP a vytvoří odpovědní dokument, který se má zaslat prohlížeči klientského počítače nazpět [6].

Existuje několik způsobů, jak může server požadavek kontejneru JSP předat. Jedním z nich je komunikace prostřednictvím zásobníku TCP/IP. Skutečnost, že serverový software i kontejner JSP jsou spuštěny na stejném počítači a komunikují spolu prostřednictvím zásobníků, není ničím neobvyklá. Tuto architekturu popisuje Obr. 4.

Při testování je možné, aby webový prohlížeč, webový server i kontejner JSP běželi na jednom počítači.



Obr. 4: **Server a klient spolu mohou komunikovat prostřednictvím protokolu TCP/IP**

3 Technologie JSP

V následujících kapitolách práce se již nebude vyskytovat popis teorie potřebné k pochopení problematiky technických norem. Další části se budou zabývat výsledkem implementace databáze norem a způsobu proč a jakými prostředky bylo kýženého úkolu dosaženo.

Webová stránka jde napsat jen s využitím jazyka HTML. Jenže jak už bylo popsáno výše v teoretické části, takováto webová stránka bude pouze statická. HTML je pouze značkovací jazyk, nemá prostředky pro zápis algoritmů.

V situacích, kdy je potřeba k požadované funkcionalitě webových stránek využít podmínku nebo přiřadit hodnotu odeslanou z nějakého formuláře do proměnné určené k dalšímu zpracování, je možné využít technologii JSP.

Ve Výpis k. 1 je uveden způsob jak je v práci v dokumentu main.jsp uživatel rozlišen na nového nebo stávajícího podle zjištění, zda má založen svůj session, nebo jestli ještě žádný nemá. Takovouto funkcionalitu pomocí HTML nelze vytvořit.

```
<% if (session.isNew()) { %>
<p> Dobrý den, vítejte v databázi norem</p>
<% } else {%>
<p> Vítejte zpět v databázi norem </p>
<%  } %>
```

Výpis k. 1: **Část dokumentu main.jsp - rozpoznání uživatele**

JSP je nástroj, pomocí kterého je možné stránkám dodat dynamiku. V následující části budou rozebrány jednotlivé možnosti JSP podrobně. Bude osvětleno, jak probíhá překlad dokumentu JSP na formu, kterou lze zobrazit pomocí webového prohlížeče.

3.1 Průběh zpracování JSP dokumentu

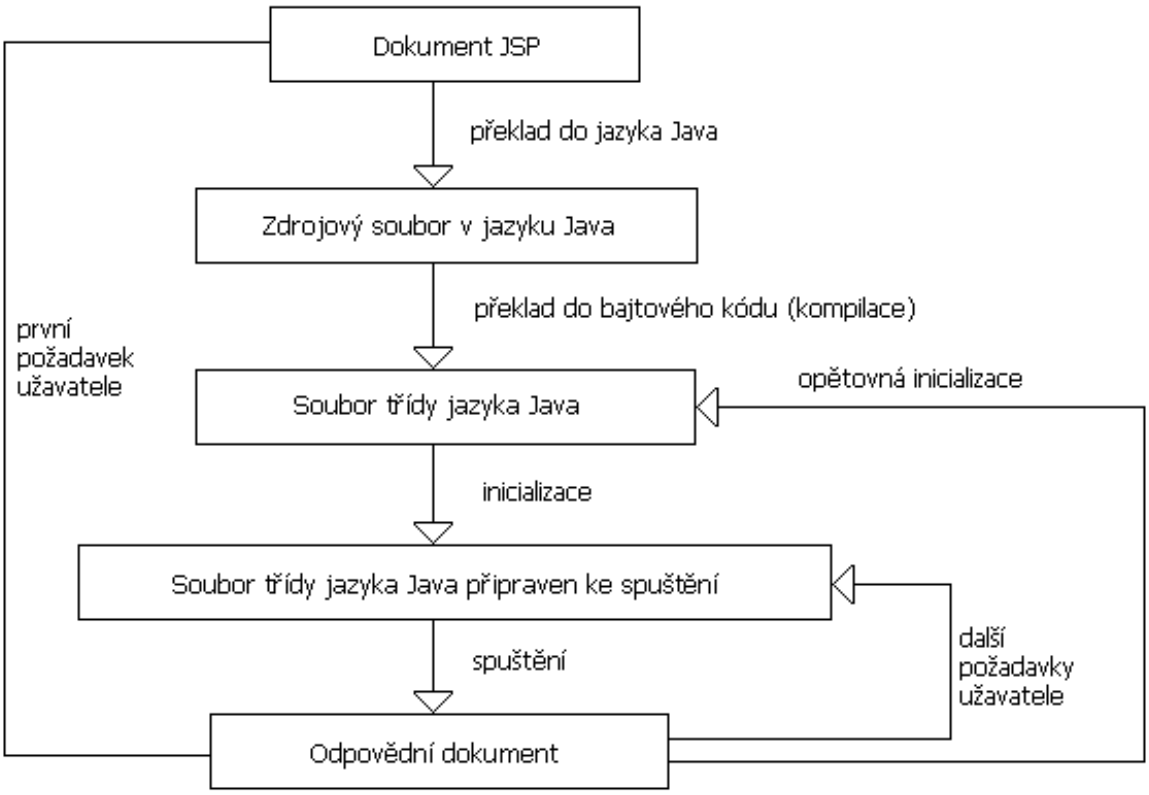
Z předchozího výpisu kódu je zřejmé, že zápisem JSP se tvoří směsice JAVA a HTML. Webový prohlížeč kód v takovéto formě zpracovávat neumí. Prohlížeč zvládá vykreslit jenom značky jazyka HTML. Je tedy nutné zajistit správný překlad z JSP na kód jazyka HTML.

Celý proces se skládá z několika dílčích částí, které budou stručně popsány. Dokument JSP je uložen v počítači, na kterém je spuštěný aplikační software, jehož součástí musí být kontejner JSP, který tyto dokumenty umožní zpracovat (například Apache Tomcat, existuje mnoho dalších).

Při příchodu prvního návštěvníka na stránku JSP je dokument kontejnerem přeložen na běžný program v Javě. Vzniká zde určitý soubor s příponou .java, který bývá označován jako servlet. Jako další krok je servlet načten překladačem jazyka Java a vzniká z nich binární soubory třídy s příponou .class. Soubory tříd obsahují všechny příkazy pro tvorbu odpovědí a pravidla jejich odeslání na klientský webový prohlížeč.

Kontejner poté načte soubor třídy .class a na jeho základě vytvoří nový objekt a nastaví proměnné. Dále se vytvoří odpovědní dokument, jenž se skládá pouze ze značek jazyka HTML a odešle se do uživatelského počítače.

Proces je poměrně časově náročný, ovšem je nutné ho provést pouze při první návštěvě dokumentu JSP. Pokud přijde znovu dotaz na stejný dokument, bude se vytvářet už pouze odpověď, ostatní části není třeba vykonávat znovu. Tuto činnost může kontejner opakovat, dokud příslušný soubor třídy není odstraněn. Dojde-li k odstranění souboru, pak musí proběhnout znovu jak práce kontejneru, tak i kompilátoru, viz Obr. 5.



Obr. 5: **Zpracování dokumentu JSP před odesláním**

3.2 Servlety

V předchozí části bylo popsáno, že servlet vznikne jako výstup z kontejneru JSP a že to v podstatě není nic jiného než program zapsaný v jazyce Java. V této části budou servlety podrobněji popsány, hlavně jejich použití a formy.

Využití servletů

Použití servletů je opravdu rozsáhlé, jsou vybrány pouze některé ilustrační případy, které byly využity v této diplomové práci.

a) Pomocí metody POST je možné přenášet z klienta data na server a tam je dále zpracovávat

b) Zprostředkovaně umožňují ukládat data do databáze pomocí JDBC

c) Vyhledávat informace ze souborů Cookies

d) Lze formátovat výstupy a počítat výsledky.

Servlety jsou založeny na principu dotaz-odpověď, typické pro komunikaci klient-server. Pro správnou komunikaci mezi klientem a serverem musí mít servlety množinu tříd definující standardní rozhraní pro obsluhu dotazů a odpovědí. Tuto problematiku řeší Java Servlet API, které se skládá ze dvou balíčků: *javax.servlet* a *javax.servlet.http*.

Standardní servlet se dědí z třídy *HttpServlet* a jeho nejpoužívanější akcí je překrytí metody *doGet* nebo *doPost*. Tyto metody zpracovávají požadavky protokolu HTTP GET a POST. Požadavky odeslané metodou GET jsou předávány v hlavičce HTTP protokolu (lze je spatřit v adresním řádku prohlížeče). Požadavky odeslané metodou POST putují uvnitř těla dotazu.

Ve Výpis k. 2 je uveden příklad servletu, který jako výsledek na klientském prohlížeči zobrazuje logovací formulář.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet
{
public void doGet (HttpServletRequest request,          HttpServletResponse
response) throws
ServletException, IOException
{
PrintWriter out;
response.setContentType("text/html");
out = response.getWriter();
        …
out.println("<h1>Prihlaseni:</h1>");
out.println("<form action="loginAction.jsp" method="POST"> ");
out.println("<div class=/"form_settings/"> ");
out.println("<p><span> Uzivatelske jmeno: </span>");
out.println("<input class=/"contact/" type=/"text/" name=/"username/"
value=username></p>");
```

```
out.println("<p><span> Heslo: </span>");
out.println("<input class=/"contact/" type=/"password/"
name=/"password/"></p>");
…
out.close();
}
}
```

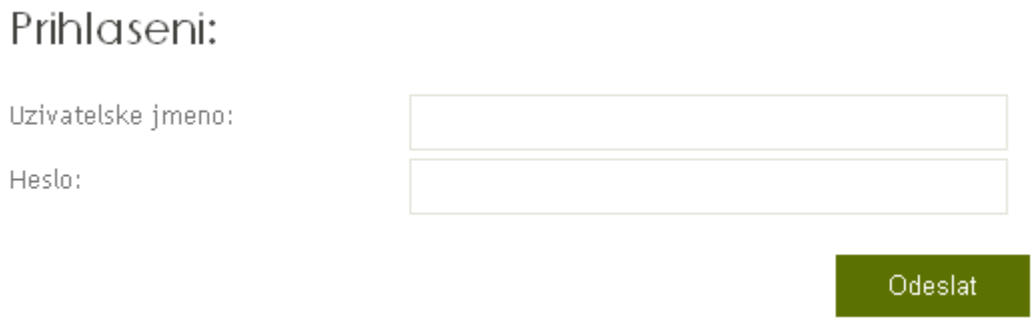
Výpis k. 2: **Část dokumentu loginForm.jsp jako servlet**

Z výpisu je patrné, že programování pomocí servletů je sice možné, ale rozhodně není vhodné pro zapisování vzhledu stránek. Snadno by docházelo k chybám, protože výstupní text je špatně rozlišitelný. Pro realizaci např. složitých výpočtů může být servlet vhodný.

```
<h1>Prihlaseni:</h1>
  <form action="loginAction.jsp" method="POST">
  <div class="form_settings">
    <p><span> Uzivatelske jmeno: </span>
      <input class="contact" type="text"
name="username"value=username></p>
  <p><span> Heslo: </span>
    <input class="contact" type="password" name="password"></p>
```

Výpis k. 3: **HTML kód z loginForm.java**

Ve Výpis k. 3 je uveden HTML kód, který vznikne spuštěným servletu z Výpis k. 2.



Obr. 6: **Zobrazení výstupu dokumentu loginForm.jsp**

V takovéto formě dorazí do klientského počítače, kde vykreslí tak, jak je zobrazeno na Obr. 6.

3.3 Způsoby zápisu JSP

V technologii JSP existují kromě kombinací Javy a HTML i další komponenty, které zajišťují chod dynamické aplikace. Jsou jimi například direktivy, akce JSP a implicitní objekty.

JSP dokumenty se skládají ze dvou částí: značek HTML a JSP příkazů. Obě tyto části jsou od sebe jednoznačně odlišeny tak, aby JSP kontejner při zpracování správně rozpoznal, co je kód zapsaný v Javě a co je HTML. Proto jsou HTML značky označeny hranatými závorkami (viz Výpis k. 4) a příkazy JSP jsou ohraničeny hranatými závorkami se znakem % (viz Výpis k. 5).

```
<b> Zadejte staré heslo: </b>
```

Výpis k. 4: **Příklad z dokumentu editUserForm.jsp**

```
<% String username = request.getParameter("username"); %>
```

Výpis k. 5: **Příklad z dokumentu loginForm.jsp**

Deklarace

JSP deklarací se rozumí zapsání jedné nebo více deklarací v jazyce Java, kde je tento kód ohraničován značkami <%! a %>. Tedy víme, že JSP deklarace se může skládat z více deklarací, proto je nezbytné, aby každá deklarace ukončila středníkem.

```
<%!
String definition= "";
    String translated_from = "";
    String term = "";
%>
```

Výpis k. 6: **Příklad deklarace v dokumentupublicGlossary.jsp**

Ve Výpis k. 6 bylo deklarováno a inicializováno několik proměnných, které budou později využity k postupnému uložení a následnému výpisů dat získaných z databáze. Proměnné jsou deklarovány jako prázdné, data budou k dispozici až po navázání spojení s databází.

Proměnné slouží k výpisu odborného termínu, jeho definice a termínu, ze kterého je přeložen ve veřejném slovníku databáze.

Výraz

Vytvoření JSP výrazu je opět velmi snadné. Pro vytvoření JSP výrazu, stačí jen výraz vepsat mezi značky <%= a %>. Zpracování příkazu probíhá tak, že server zjistí, jakou hodnotu má daný příkaz a tuto hodnotu zobrazí v klientském počítači.

```
<%=term%>
<%=definition%>
<%=translation_from%>
```

Výpis k. 7: **Příklad výrazu v dokumentu publicGlossary.jsp**

Výpis k. *7* slouží k vypsání hodnoty, která je v jednotlivých proměnných uložena. Kontejner JSP sám zjistí, o jaký typ proměnné se jedná a zvolí vhodnou metodu pro výpis. V tomto případě se vypisuje obsah proměnných typu String. Proměnné již mají v sobě uložena data získaná z databáze, nyní se jen vypisují na obrazovku. Komunikaci s databází se podrobně věnuje kapitola 4.

Skriptlety

Skriptlet je všestranný element JSP, který je v dokumentu JSP ohraničen dvojicí značek <% a %>. Jedná se opravdu výkonný element, jelikož do skriptletu je možné napsat libovolný kód jazyka Java. Tohoto elementu je v práci hojně využíváno, jedná se o velmi univerzální vývojový prostředek. Velkou výhodou je, že skriplet nemusí být celistvý a funkčně soběstačný.

```
<% while(resultset.next()) {
language=resultset.getString("LANGUAGE");%>
<%=language%><br>
<% } %>
```

Výpis k. 8: **Příklad části dokumentu userStandards.jsp**

Ve Výpis k. 8 jsou dva skriptlety ohraničené značkami <% a %> a také HTML *tag* <*br*> pro odřádkování. První skriplet obsahuje cyklus *while,* který z datové struktury *resultset* postupně získává data, která ukládá do proměnné *language*. Data jsou poté vypisována pomocí výrazu `<%=language%>`, jak bylo popsáno výše. Druhý skriplet obsahuje pouze závorku, která

ukončuje tělo cyklu *while*. Tímto způsobem je zajištěno opakované vykonání těla cyklu včetně HTML tagů. Praktickým výsledkem činnosti je výpis jazyků použitých v normách, které jsou uloženy v databázi.

V případě, že jsou deklarace prováděné ve skriptletech, pak je proměnná inicializována při každém dotazu na stránku. Naopak deklarace JSP vytvoří proměnnou a nastaví její hodnotu již během inicializace dokumentu. Proto je potřeba pečlivě uvážit, v kterých situacích použít jakou deklaraci v závislosti na použití při zohledněním faktu, že inicializace jistou dobu trvá.

	Deklarace	Skriptlet	Výraz
Uzavřeno v	<%! %>	<% %>	<%= %>
Obsahuje	Jedna nebo i více deklarací v jazyce Java	Kód v jazyce Java mezi značky lze zadat buď úplné příkazy, nebo i pouze ústřižky, které mají pokračování mezi dalšími značkami.	Jeden výraz v jazyce Java
Účel	Vytvoří název, ke kterému lze přiřadit i počáteční hodnotu	Sděluje systému, aby provedl příkazy, které jsou mezi značkami	Vrací hodnotu proměnné
Spuštěno	Při první návštěvě stránky nějakým klientem nebo v okamžiku kdy kontejner JSP opětovně inicializuje stránku	Vždy když nějaký klient navštíví stránku	Vždy když nějaký klient navštíví stránku

Tab. 2: **Porovnání možných zápisů**

3.4 Implicitní objekty

Implicitní objekty jsou dalším z nástrojů vývojáře. Jméno implicitních objektů je odvozeno z toho, že je lze používat, aniž by bylo nutné je nějak nadefinovat. Objekty vznikají vždy jako součást procesu překladu JSP dokumentu na servlet. Zároveň jsou dostupné v jakýchkoli výrazech a skriptletech použitých v dokumentu JSP. V JSP každý implicitní objekt odpovídá určité třídě nebo rozhraní Javy.

Objekt typu request

Objekt *request* poskytuje metody díky, kterým je možné zpracovávat data zaslaná klientem pomocí webového požadavku na server. Výpis k. 9 ilustruje možnost získání hodnoty *language* odeslané ze stránky postForm.jsp pomocí formuláře využitím metody POST protokolu HTTP. Je to způsob jak předat a posléze zaznamenat do databáze jazyk, ve kterém je norma napsána.

```
<% String language = request.getParameter("language");%>
```

Výpis k. 9: **Příklad části dokumentu z postCheck.jsp**

Objekt typu response

Při přijetí dotazu od klienta na server se automaticky skládá a odesílá odpověď na dotaz. Kontejner JSP uloží informace o této odpovědi právě do objektu typu *response*. Vlastnosti této odpovědi je možné měnit pomocí metod, které nám objekt poskytuje.

```
response.setContentType("image/jpg");
```

Výpis k. 10: **Příklad části dokumentu captcha.jsp**

Pomocí části kódu (viz Výpis k. 10) je realizováno, že odpověď z dokumentu captcha.jsp bude obrázek nikoli text. Této vlastnosti je využito při registraci nového uživatele, kdy výsledek z dokumentu captcha.jsp je interpretován ve formuláři jako obrázek k přepsání. Vlastnost slouží k rozpoznání skutečných uživatelů od robotů, o této problematice podrobně pojednává kapitola 5.1.

Objekt typu out

Implicitní objekt typu *out* je instancí třídy *JspWriter* a poskytuje především dvě významné metody *out.print()* a *out.println()*. Voláním těchto metod se text odesílá do webového dokumentu. Rozdíl mezi těmito dvěma metodami je, že metoda *println()* na konci řetězce odřádkuje.

Objekt *out* je možné využít k tomu, aby nebylo zapotřebí dělit skriptlety na mnoho malých částí. Na funkci to nemá žádný vliv, ale kód je potom přehlednější pro vývojáře. Příklad Výpis k. 11 ilustruje použití implicitního objektu *out*.

```
<% while(resultset.next()) {
language=resultset.getString("LANGUAGE");
out.print(<% = language %>);
 } %>
```

Výpis k. 11: **Příklad použití objektu out v dokumentu publicGlossary.jsp**

Objekt typu session

Implicitní objekt typu *session* je vytvořen kontejnerem JSP v případě, kdy na server přijde požadavek nového klienta. Dále při každé další návštěvě už kontejner používá informace ze *session* vytvořené při prvním dotazu klienta na server.

Ke každé *session* lze přiřadit atributy, které se pak ukládají jako soubory *cookie* na pevný disk uživatele. Vývojáři přistupují k těmto atributům pomocí metod objektu *session*. Pro nastavení slouží metoda *setAttribute()* a pro získání dat nazpět slouží metoda *getAttribute()* objektu *session*.

Implicitní objekt *session* obsahuje několik zajímavých metod. Jejich výběr je znázorněn v Tab. 3.

Metoda	Význam
getCreationTime()	Vrací čas vzniku uživatelské relace
getLastAccessedTime()	Vrací okamžik poslední návštěvy daného uživatele
getMaxInactiveInterval()	Vrací časový údaj, po jehož uplynutí je relace ukončena. (časový limit nečinnosti)
setMaxInactiveInterval()	Umožňuje nastavit časový limit nečinnosti, v práci nastaveno na 60*30 sekund.
getAtrtibute	Vrací objekt shodný s názvem parametru. Pokud zadaný název atributu neexistuje, pak vrací hodnotu *null*
Void setAttribute	Přiřadí objekt k dané relaci pod zadaným názvem

Tab. 3: **Výčet metod objektu session**

Bez objektu session by se praktická realizace těžko obešla. Pomocí cookie uložené na uživatelově počítači je realizováno celé přihlašování uživatelů. Bez přihlašování by databázi pravděpodobně nešlo plnohodnotně vytvořit.

```
<%String username =
(String)session.getAttribute("username");
session.setMaxInactiveInterval(60*30); %>
```

Výpis k. 12: **Příklad využití objektu session v main.jsp**

Výpis k. 12 zajistí zjištění uživatelského jména ze souboru cookie, uživatelské jméno do session bylo uloženo dokumentem loginAction.jsp. Nyní ho lze použít k zobrazení jména přihlášeného uživatele. Metoda setMaxInactiveInterval() nastavuje maximální délku nečinnosti uživatele na 30 minut, poté platnost session zaniká a uživatel se bude muset znovu přihlásit.

3.5 Direktivy JSP

Direktivy na rozdíl od ostatních elementů neobsahují žádný kód v jazyce Java. Direktivy JSP si lze představit jako zprávy, které jsou zaslány kontejneru, tím se žádá o vyřízení nějaké akce v dané stránce.

Direktivy je možné nadefinovat takovýmto způsobem:

```
<%@page contentType = "text/html;charset=windows-1250" %>
```

Výpis k. 13: **Direktiva page ze souboru thighToUseAndImport.jsp**

Direktiva je ohraničena značkami <%@ a %>. Po počáteční značce následuje její název.

K dispozici jsou tři druhy direktiv JSP: *include*, *page* a *taglib*. Nyní budou dvě direktivy v krátkosti popsány. Direktiva t*aglib* popsána není, protože jí v práci nebylo využito.

Direktiva include

Direktiva *include* je velice důležitým nástrojem JSP. Poskytuje možnost, pomocí níž lze dosáhnout především usnadnění práce a zpřehlednění kódu.

Vývojář stránek často naráží na problém, že v dokumentech JSP se často opakují stejné pasáže kódu.

Právě v této situaci je vhodné využít direktivy *include*. Direktiva *include* má pouze jeden atribut, a to je *file*. Jako hodnota atributu file, se udává název dokumentu JSP, který chceme do příslušného dokumentu vložit prostřednictvím místo direktivy *include*.

```
<%@ include file="thingsToUseAndImport.jsp" %>
```

Výpis k. 14: **Direktiva include využitá v každé stránce aplikacce**

V praktické části této diplomové práce je direktivy *include* využíváno velmi často. Její pomocí jsou do každého dokumentu vložena důležitá data. Především se jedná o napojení na JavaBeans (viz kapitola 3.6) a nastavení znakové sady v dokumentech použité.

Direktiva page

Direktiva *page* je velmi obecná direktiva. Obsahuje velký počet atributů, které upravují různé vlastnosti stránek.

Vybrané atributy direktivy:

- *autoFlush*

- *buffer*
- *contentType*
- *errorPage a isErrorPage*
- *extense*
- *import*
- *info*
- *isThreadSafe*
- *language*
- *session*

Atribut *import* direktivy *page* slouží ke zkrácení zápisu jednotlivých tříd ve skriptletech na dané stránce. Dá se též říci, že pomocí atributu *import* nahrajeme potřebné knihovny. Ve Výpis k. 15 si ukážeme využití atributu *import* direktivy *page* na reálném příkladu.

```
<%@ page import="java.io.*"%>
<%@ page import="java.awt.image.*"%>
```

Výpis k. 15: **Výpis importů potřebných pro funkci dokumentu captcha.jsp**

Je-li část kódu přítomna na počátku dokumentu JSP, pak je možno v tomto dokumentu využívat veškerých metod z balíčků, které importy obsahují.

Balíček *java.awt.image* obsahuje různé metody pro práci s obrázky, *java.io* umožňuje využívat různých nástrojů pro operace se vstupem a výstupem, tyto balíčky jsou pro vytvoření obrázku captcha nutné.

3.6 Objektový model JavaBeans

Prostředí Java Server Pages poskytuje několik standardních akcí. Každou lze považovat za samostatnou programovací techniku. V této podkapitole bude popsán objektový model JavaBeans. Programováním objektů modelu JavaBeans nelze získat žádný dodatečný výkon. Vše co lze udělat v prostředí Java Server Pages a pomocí webového serveru, lze udělat i bez objektů JavaBeans. Přesto je vhodné JavaBeans zauvažovat při objektových souvislostech, což učiní programy přehlednějšími. Programy jsou lépe organizovány, snáze se testují, ladí, udržují a případné dodatečné změny jsou snáze proveditelné. Dále je méně náročné jejich znovu použití v dalších projektech.

JavaBeans jsou odpovědí firmy Sun Microsystems na technologii Microsoft ASP. Objekt JavaBeans je vlastně Třídou jazyka Java, jejíž charakteristika a chování jsou dostupné všem zbývajícím částem kódu. Termín dostupné znamená, že vnější kód může automaticky rozpoznat metody odpovědné za manipulaci s proměnou. Popisované automatické rozpoznání je ale závislé nejen na samotné proměnné, ale i na tom, zda se názvy jejích metod řídí standardy pojmenování modelu JavaBeans.

Každá vlastnost je název proměnné začínající malým písmenem. Každá průvodní metoda začíná prefixem *set*, *get* nebo *is*, za nímž následuje název proměnné začínající velkým písmenem.

Každá průvodní metoda může být buď metodou změny (*setter method*), nebo metodou čtení (*getter* method). Průvodní metody čtení mají prefix *get*. V případě, že metoda čtení vrací booleovskou hodnotu, je přípustné, aby její název začínal prefixem *is*.

Průvodní metody jednotlivých vlastností jsou veřejné. Proměnné uvnitř jsou ale deklarovány jako soukromé. Objekty modelu JavaBeans si vynucují úpravy a čtení svých vlastností bezpečným a předpověditelným způsobem. Obsah jednotlivých vlastností mohou měnit i další metody, ovšem stále pouze bezpečným způsobem. Objekty ovšem nemusí obsahovat pouze průvodní metody vlastností, dále všechny vlastnosti nemusí mít obě průvodní metody (*set* a *get*).

Při pohledu do zápisů servletů lze zjistit, že HTML značky jsou vnořeny do kódu jazyka Java. Takovýto servlet je sice funkční, ovšem jeho zápis rozhodně není elegantní a přehledný. Ideální je, když se podaří izolovat značky HTML od kódu v jazyce Java. Dokumenty s jasně oddělenými bloky kódu jsou nejen přehlednější, ale snazší je i jejich správa. Kód vložený

do souboru s příponou .java a v příslušném dokumentu JSP je pouze odkaz na příslušný soubor. Právě to je základní filosofie užití objektů modelu JavaBeans v dokumentech JSP [6].

Výpis k. 16 demonstruje dokument obsahující instrukce useBean. Výpis k. 17 pak ukazuje definici objektu využitého prostřednictvím zmíněné instrukce.

```
<%@page contentType = "text/html;charset=windows-1250" %>
<% request.setCharacterEncoding("windows-1250"); %>

<jsp:useBean id="db"
        class="bb.messageBoard.DbAccess"
        scope="application"/>

<jsp:useBean id="work"
        class="bb.messageBoard.Worker"
        scope="application"/>

<%@page import="java.sql.*,
          java.text.DateFormat,
          java.util.Vector,
          java.io.*"%>
```

Výpis k. 16: **JavaBeans v dokumentu thingsToUseAndImport.jsp**

```
package bb.messageBoard;
import java.sql.*;

public class DbAccess implements java.io.Serializable {
   private Connection connection;
   private Statement statement;
   public DbAccess ()
        throws ClassNotFoundException, SQLException
   {
```

```
            Class.forName("com.mysql.jdbc.Driver");
        connection = DriverManager.getConnection ("jdbc:mysql://localhost:3306/norm", "root",
"root");
            connection.setAutoCommit(false);
            statement = connection.createStatement();
          }
        public void executeUpdate(String sqlCommand) throws SQLException
          {
            statement.executeUpdate(sqlCommand);
          }
        }
```

Výpis k. 17: **Výpis části třídy DbAccess.java použité jako JavaBean**

Instrukce useBean vytvoří instanci třídy *DbAccess*. Třída má nadefinovaný konstruktor, který vytvoří spojení s databází. Poté se při volání metody executeUpdate(), kterou lze díky vlastnostem modelu JavaBeans zavolat z kteréhokoli dokumentu aplikace, vykoná příkaz jazyka SQL, který je předán jako parametr proměnné. Scénář pro druhého návštěvníka dokumentu je částečně odlišný. Tentokrát už kontejner JSP obsahuje instanci třídy *DbAccess*. Není tedy třeba znovu vytvořit další. Při vykonávání akce specifikované instrukcí *useBean* je vyhledána existující instance třídy *DbAccess*. Instance je následně zpřístupněna i druhému návštěvníkovi. Efektivnost objektu modelu JavaBeans závisí na tom, zda kontejner JSP najde existující instanci příslušné třídy pokaždé, když stránku navštíví nový uživatel. To, že instance objektu trvá i mezi uživatelskými relacemi určuje atribut *scope* instrukce *useBean* (viz Výpis k. 16), jehož nastavení určuje platnost objektu na dobu trvání relace dané aplikace. Objekt s rozsahem platnosti na úrovni aplikace je dostupný všem, kteří navštíví stránku během relace aplikace.

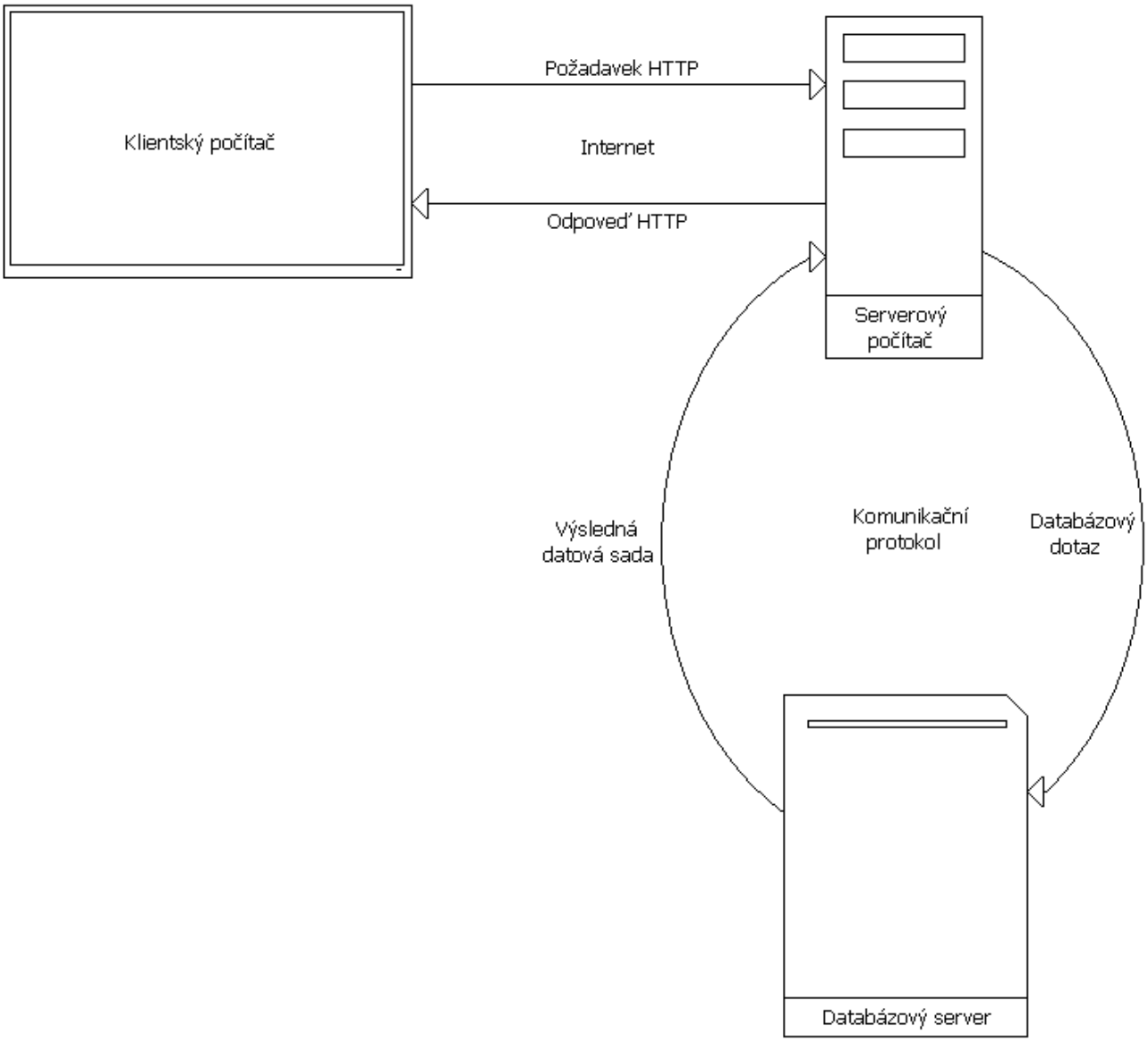
Při práci s instrukcí *useBean* se dokument JSP snaží nejprve najít existující instanci objektu. Nenajde-li ji, vytvoří instanci novou. Jakmile je instance třídy vytvořena, lze její metody volat ve všech skriptletech, jež jsou v rozsahu platnosti vytvořené instance. K volání průvodních metod jednotlivých vlastností použitého objektu lze kromě standardní syntaxe použít i instrukce *setProperty* a *getProperty*.

4 JSP a databáze

Možnost propojení JSP s nějakou databází pro vývoj dynamické aplikace je v podstatě nutností. Mnoho dynamických webových aplikací zpracovává velké množství dat, která jsou ukládána do databáze, proto je komunikace s ní důležitou součástí JSP. V praktické části této diplomové práce je databáze také využívána. Do databáze jsou ukládána veškerá data získaná od uživatelů.

4.1 Komunikace s databázovým serverem

Při příchodu klientského dotazu na webový server se dotaz vyřizuje standardním způsobem. Webový server spustí příslušný soubor *.class*, jenž vznikne překladem stránky JSP. Během zpracování dotazu serverový počítač narazí na dotaz, který se odvolává na databázi. Serverový počítač předá příslušná volání softwaru, který je uložen na databázovém serveru (komunikace probíhá pomocí protokolu TCP/IP). Databázový server požadavek zpracuje, vytvoří výstupní sadu a odešle ji zpět na počítač, kde běží aplikační server, ten sadu zpracuje a na klientský počítač odešle svoji odpověď (viz Obr. 7).



Obr. 7: **Komunikace databáze-server**

4.2 MySQL

MySQL je velmi populární a dnes často využívaná databáze, podle mnoha zdrojů je rovněž velmi rychlá. Nemá však tolik funkcí a možností jako některé konkurenční databázové systémy (*PostgreSQL*). MySQL je multiplatformní, to znamená, že je dostupná pro více počítačových platforem. Komunikace probíhá za pomoci jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s jistými rozšířeními [17].

Pro napojení Javy k MySQL databázi použijeme JDBC (Java Database Connectivity). JDBC je vrstva mezi Java aplikací a vlastní komunikací s databází. Datové typy SQL lze získat

z výsledku SQL dotazu jako instance Java tříd a s těmi pracuje aplikace. Naopak JDBC ovladač (driver) dokáže vkládat instance Java tříd do SQL dotazů a správně je uložit, upravit apod, v závislosti na zvolené databázi. Takto lze vytvořit aplikaci nezávislou na zvoleném databázovém stroji. Nicméně je stále potřeba psát dotazy v SQL.

K přístupu k MySQL je nutné nainstalovat ovladač této databáze, jeho aktuální verzi lze najít na oficiálních stránkách MySQL. Předtím, než budou provedeny jakékoli operace nad databází, je nutné ovladač nahrát a registrovat. K tomu slouží příkaz *Class.forName*. Jakmile byl JDBC ovladač nahrán a správně zaregistrován *DriverManagerem*, je možné navázat spojení s databází. *DriveManager* nám poskytuje metodu *getConnection*, která nás připojí k příslušné databázi. Jakmile máme spojení s databází, můžeme nad ní začít provádět datové operace. K tomu je určena další třída *Statement*, která nám poskytne vytvořený objekt *Connection*. Instance této třídy pak slouží pro posílání SQL příkazů databázi, kde jsou zpracovány a vrací odpovídající hodnoty [18].

```
package databaze;
import java.sql.*;
public class Databaze {
	public static void main(String[] args) {
		try{
			Class.forName(”com.mysql.jdbc.Driver”);
		}
		catch(Exception e){
			System.out.println(”Chyba driveru”);
			System.out.println(e.toString());
		}

		try{
			Connection conn =
			DriverManager.getConnection(“jdbc:mysql://localhost:3306/data”,”root”,”passwd”);
			Statement st = conn.createStatement();
			ResultSet rs = st.executeQuery(”SELECT * FROM table”);

			while (rs.next()) {
				System.out.println(rs.getString(1));
			}
		}
		catch(Exception e){
			System.out.println(”Chyba volani”);
			System.out.println(e.toString());
		}
	}
}
```

Výpis k. 18: **Připojení k databázi**

4.3 Aplikační server Apache Tomcat

Apache Tomcat je jedním z nejvýznamnějších aplikačních serverů, vyvíjen jako open source projekt. Je založen na jazyce Java, javových servletech, JSP a EJB (Enterprise JavaBeans). Je to vlastně kontejner JSP, který obsahuje nástroje pro správu a management. Je také možné ho zpravovat na dálku prostřednictvím XML konfiguračních souborů.

Po instalaci Tomcatu se do proměnné CATALINA_HOME přidá hodnota odpovídající jeho domovské adrese v systému adresářů. V tomto adresáři se nacházejí další důležité podadresáře. Jejich výpis a použití je popsáno v Tab. 4 [13].

/bin	Obsahuje spustitelné soubory samotného aplikačního serveru
/common/lib	Obsahuje JAR knihovny pro webové aplikace a interní kód Tomcatu
/conf	Obsahuje konfigurační soubory
/logs	Obsahuje soubory logů jednotlivých aplikací
/shared/lib	Obsahuje JAR knihovny sdílené mezi webové aplikace
/webapps	Domovský adresář všech hostovaných aplikací

Tab. 4: **Významné adresáře aplikačního serveru**

4.4 Výsledná databáze

K vytvoření databáze technických norem a jejich terminologie podle požadavků popsaných v kapitole 1, byla navržena následující databázová struktura:

a) users - Tabulka má 4 sloupce, její obsah shrnuje informace dostupné o uživatelích:
    a. USERNAME – Obsahuje uživatelské jméno, zároveň slouží i jako primární klíč, protože je vhodné, aby uživatelské jméno bylo v systému unikátní. Tím se ušetřil jeden sloupec tabulky.
    b. PASSWORD – Obsahuje zašifrované a osolené heslo. Problematikou utajení hesla se podrobně zabývá kapitola 5.2.
    c. LANGUAGE – Obsahuje jazyk, kterým má aplikace s uživatelem komunikovat.
    d. EMAIL – Obsahuje E-mail který zadal uživatel při registraci.
b) messages – Tabulka obsahuje 10 sloupců, její obsah shrnuje informace o normách:
    a. WHENMADE – Primární klíč tabulky, obsahuje desetimístný unikátní identifikátor.
    b. USERNAME – Jméno uživatele, který danou normu do systému vložil.
    c. SIGNIFICATION – Označení normy.

d. NAME – Jméno normy v jazyce, ve kterém je napsána.
e. ORGANIZATION – Jméno organizace, která technickou normu vydala.
f. DATEOFISSE – Datum vydání technické normy.
g. REPLACES – Označení normy, kterou daná norma nahrazuje.
h. NUMBER – Číselné označení normy.
i. ALTERNATIVNAME – Označení normy v jazyce jejích originálu.
j. LANGUAGE – Jazyk normy, ve kterém je napsána.

c) glossary - Tabulka obsahuje 5 sloupců, její obsah shrnuje informace o uložené terminologii:
   a. IDENTIFIER - Primární klíč tabulky, obsahuje desetimístný unikátní identifikátor.
   b. TERM – Odborný termín, ke kterému se vztahuje definice.
   c. DEFINITION – Slovní definice daného termínu.
   d. TRANSLATION_FROM – Termín v jazyce, ze kterého byla norma přeložena.
   e. STANDARD_ID - Desetimístný unikátní identifikátor normy, ke které se termín vztahuje.

5 Použitá řešení

5.1 CAPTCHA

Mnohé ze služeb, které jsou v dnešní době provozovány na Internetu, by se neobešly bez tzv. plně automatizovaného Turingova testu (*completely automated public Turing test to tell computers and humans apart*). Test je nazýván podle počátečních písmen jeho názvu zkráceně „captcha“.

Když v roce 1950 známý anglický matematik Alan Turing vymyslel test, který je dodnes označován jako „Turingův test“, žádný Internet ještě neexistoval. I vyspělost tehdejší techniky byla zcela jiná než dnes, přesto už tehdy se lidé zabývali otázkou, která je současně o to aktuálnější.

I když může znít poněkud absurdně: „je někdo nebo něco na druhé straně člověk nebo stroj?“ V tehdejší době byla otázka spíše akademická, v souvislosti s prvními krůčky oboru, usilujícího o vytvoření umělé inteligence.

Právě tuto otázku odpovídá Turingův test, který nechává umělou inteligenci předstírat, že je člověkem, a další lidé pak mají za úkol posoudit, zda opravdu jednají s člověkem či se strojem. Pokud nejsou schopni odpovědět, nebo odpovídají špatně, pak se stroj úspěšně vydává za člověka a lze prohlásit, že disponuje umělou inteligencí.

Do dnešní doby stále žádná umělá inteligence, která by prošla Turingovým testem, nebyla vytvořena. Ovšem v prostředí Internetu dnes najdeme roboty softwarové, kteří sami pracují na počítačích, či získávají informace v prostředí celosvětového Internetu.

A právě u různých skriptů, robotů, programů, či kódů, působících v prostředí Internetu, vzniká problém jiný, opačný. Již velmi jednoduchý program může provádět činnosti, jako je registrace u nějaké zdarma poskytované webové služby a následně ji začít zneužívat. Může si například vytvořit účet na nějakém freemailovém serveru a začít odesílat spam, nebo vyplnit komentář v diskusním serveru.

Možností takovýchto programů lze ale snadno zneužít. Třeba k zahlcení nějaké on-line diskuse strojově vkládanými komentáři, které často obsahují nějaký druh reklamy. Nebo k hromadnému vytváření nových účtů na nějaké zdarma poskytované službě, čímž jejímu

poskytovateli vznikají náklady, a může dojít i k zahlcení služby jako takové. K odstranění tohoto jevu je potřeba něco jako Turingův test obráceně.

Většina takových protiopatření přitom staví na snaze rozlišit, zda požadavek přichází od člověka nebo od stroje. Požadavek na poskytnutí služby je poté akceptován, pouze pokud test vyhodnotí, že přichází od člověka. Jde tedy o analogii Turingova testu, ovšem vlastně obráceně. Už nejde o pozitivní test, který má někoho někam přijmout, ale spíše o negativní test, který má někoho vyloučit.

Také techniky, které se v tomto případě používají, sází spíše na omezenost schopností robota, než aby testovaly jejich dostatečně vysokou úroveň. Záměrně totiž využívají něčeho, co je pro člověka relativně snadné a bezproblémové, zatímco pro robota (obvykle relativně jednoduchého) je nepřekonatelným problémem. Tyto technicky samotné však musí být jednoduché a schopné autonomního fungování, tak aby nevyžadovaly přímou součinnost člověka.

Techniky usilující o rozpoznání lidského uživatele od stroje fungují nejčastěji formou *Captcha*. Jsou to obrázky, na kterých je nějaký "pokroucený" text nebo víceciferné číslo, navíc s různě komplikovaným pozadím, které znesnadňuje jeho čitelnost. Nikoli tak moc, aby to člověk už nemohl přečíst. Ovšem už dost nečitelné na to, aby selhaly případné snahy robotů o automatické vyplnění formuláře.

Obr. 8: **Captcha**

Na Obr. 8 je vyobrazena Captcha použitá v této práci, z důvodu snadnějšího rozpoznání pro uživatele obsahuje pouze čísla. V HTML formuláři vystupuje jako JPG soubor, potencionální uživatel musí přepsat číslice do připraveného formuláře, jinak nebude jeho registrace platná. Tímto se lze vyhnout vkládání nesmyslných textů do databáze a další vandalizaci celé aplikace.

Kromě rozpoznání textu připadají v úvahu i další formy kontroly, jako třeba různé slovní úlohy, na které musí uživatel správně odpovědět. Zvolená forma kontroly byla ale shledána jako dostatečná pro svůj účel.

5.2 Zabezpečení hesel

V práci jsou hesla uložená do databáze zahashované. Útočník, který by je nějakým způsobem z databáze získal, tak neuvidí jejich původní verze, ale pouze jejich otisky (hashe). Zabezpečení je ovšem možné oslabit při použití různých metod. Bezpečnost takto uložených hesel se pro takové případy dá navýšit takzvaným „solením“.

Pokud se někdo z venčí dostane k hashovaným heslům, může se pokusit z daných otisků získat původní hesla. To jde s jistotou provést pouze útokem hrubou silou, kdy se generuje systematicky jedno možné heslo za druhým, z každého se spočítá *hash* a porovnává s lámaným otiskem. Takový postup je výpočetně velmi náročný. Existují ale další způsoby, jak rozlomení hesla urychlit.

Prvním je slovníkový útok, který staví na tom, že původní heslo je postavené z existujících slov běžného jazyka, případně na konci doplněných o číslo. Množství zkoušených kombinací pro porovnání se tak radikálně sníží.

Poslední metoda spočívá ve využití *rainbow tables*. Jedná se o předem vypočítané tabulky dvojic „heslo – otisk“. Pro zkoumaný *hash* pak v tabulkách pouze stačí najít odpovídající původní heslo. Takové prolomení je opravdu relativně rychlé. *Rainbow tables* jsou náročné na úložný prostor, proto se zpravidla opět počítají jen z omezené skupiny hesel, například ze slovníku nějakého jazyka.

Další hrozbou u jednoduše hashovaných hesel je fakt, že všichni uživatelé se stejným heslem mají i stejný *hash*. Pokud se tedy podaří určit z nějakého otisku původní heslo, získává přístup ke všem dalším uživatelům se shodným heslem.

Většinu z uvedených problému řeší tzv. „solení“ hesel. Při hashování hesla se na vstup navíc ještě přidává další řetězec znaků, nazývá se sůl. Sůl musí být pokaždé jiná, výsledný hash je tak i pro stejná hesla odlišný.

Jako sůl může být obecně cokoliv, dokonce ani není nutné, aby byla tajná. Naopak, často jde o veřejnou hodnotu. Jediný požadavek na sůl je, aby se lišila navzájem mezi jednotlivými uživateli.

Nejsnadnějším řešením se může zdát sůl automaticky generovaná. Tu je ale nutné v databázi uložit do speciálního sloupce. Výrazně praktičtější je proto jako sůl využít samotného uživatelského jména. Ušetří se jeden databázový sloupec. Není nutné ho někam posílat,

uživatelské jméno je rovnou k dispozici. Druhý způsob je použit v této práci. Před počítáním *hash*e je za heslo přidáno konkrétní uživatelské jméno. Teprve pak probíhá samotný proces šifrování hesla.

```
MessageDigest alg = MessageDigest.getInstance("MD5");
   alg.reset();
   alg.update(hashedString.getBytes());
   byte[] digest = alg.digest();

   StringBuffer hashedstring = new StringBuffer();
   String hx;
   for (int i=0;i<digest.length;i++){
         hx =  Integer.toHexString(0xFF & digest[i]);
         if(hx.length() == 1){hx = "0" + hx;}
         hashedString.append(hx);
   }
   String hashedString = hashedString.toString();
```

Výpis k. 19: **Výpočet hashe**

Osolená hesla jsou odolná před útoky s před-generovanými tabulkami (rainbow tables). Protože by si útočník musel vygenerovat speciálně tabulku pro tuto aplikaci, což náročnost úkolu posouvá na úroveň útoku hrubou silou.

Obtížnost prolomení hesel při útoku hrubou silou lze zvýšit zejména pomocí požadavků na minimální délku a sílu zadávaných hesel. V práci vyřešeno tak, že aplikace vyžaduje heslo alespoň délky 6 znaků, které není shodné s uživatelským jménem.

6 Instalace Aplikace

K úspěšnému oživení výsledné aplikace, která je jako příloha této práce dodána na CD je nutné nainstalovat několik programů na hostujícím počítači (serveru):

a) Java Runtime Enviroment (JRE) - je prostředí pro běh aplikací vyvinutých v jazyce Java. Prostředí pro požadovaný operační systém lze stáhnout ze stránek [12].

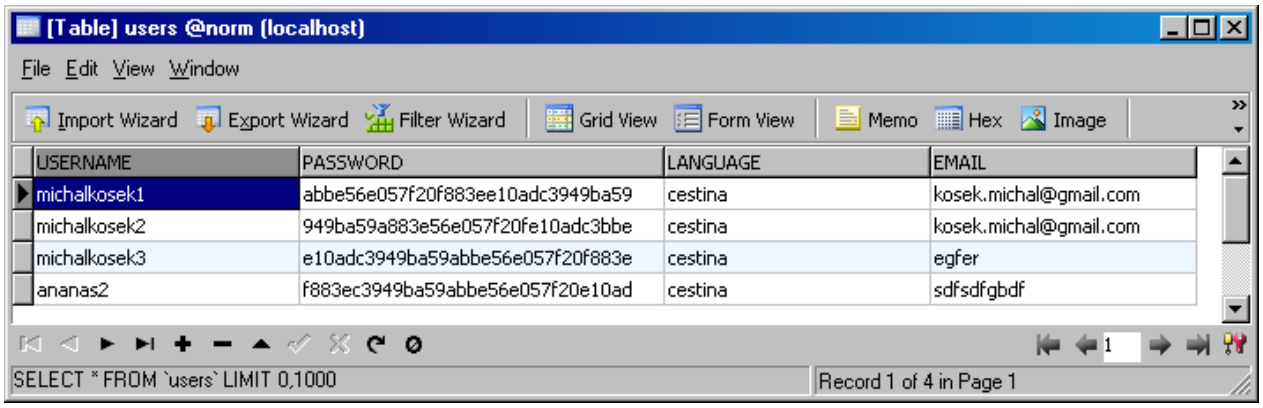
b) Databáze MySQL - další krok spočívá v instalaci databázový software. V této práci byla použita databáze *MySql*, kterou je možné stáhnout ze stránek [9]. Po úspěšné instalaci databáze je nutné v aplikaci v souborech JavaBeans s názvem *DdAccess.java* umístěných v balíčku *bb.messageBoard* nakonfigurovat adresu databáze, uživatelské jméno a heslo, tak jak je to zobrazeno na Výpis k. 20. Důvodem je že databáze a hostitelský počítač mohou být jiná zařízení. I v případě běhu na stejném počítači je nutné, aby se tyto dvě vrstvy správně spojily.

```
connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/norm", "root", "root");
```

Výpis k. 20: **Konfigurace databázového spojení**

V nově nainstalované, tedy zcela čisté databázi je nutné vytvořit jednotlivé tabulky a sloupce přesně podle toho, jak je očekává aplikace. To lze realizovat s využitím příkazového řádku, kdy jsou postupně jednotlivé sloupce vytvořeny pomocí SQL příkazu create, postup je zdlouhavý a navíc je pravděpodobné, že se vyskytne překlep v přepisu. Z tohoto důvodu je v balíčku bb.messageBoard přiložen soubor CreateMessageBoardTables.java. Ten obsahuje svojí vlastní metodu main(). Díky tomu, může být spuštěn samostatně nezávisle na zbytku aplikace. Soubor se postará o vytvoření základu databáze zcela sám, to znamená, že založí jednotlivé potřebné tabulky v databázi včetně všech potřebných sloupců.

Do databáze MySQL lze nahlížet i nějakým specializovaným nástrojem, který má grafický výstup. Tím je možné snadno celou aplikaci ladit případně provádět ruční editace a kontrolu záznamů (viz Obr. 9).



Obr. 9: **Navicat for MySQL - grafické rozhraní pro práci s databází**

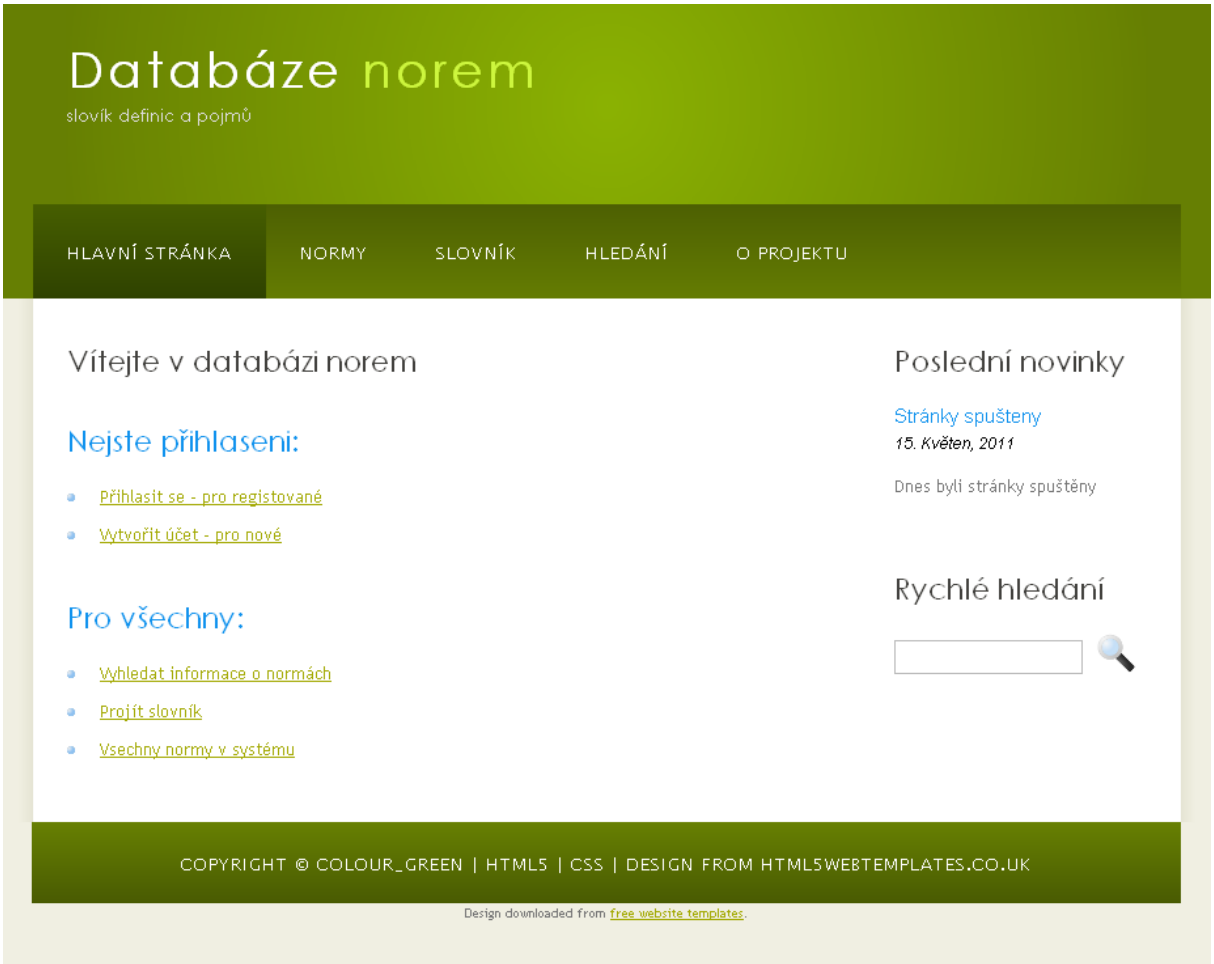
c) Apache Tomcat - poslední část oživení spočívá v instalaci aplikačního serveru. Pro tuto aplikaci byl zvolen Apache Tomcat, který lze stáhnout ze stránek[13].

Po jeho instalaci již stačí soubory aplikace nakopírovat do složky /webapps, kterou lze nalézt v jeho kořenovém adresáři.

Nyní po zadání adresy tohoto počítače do prohlížeče by měl klientský počítač zobrazit úvodní stránku aplikace.

7 Uživatelská příručka

Aplikace, která je výsledkem této práce umožňuje svým návštěvníkům vícero možností. V případě, že aplikaci navštíví uživatel, který nemá platnou session, to znamená, že se na stránky v daném prohlížeči za dobu posledních třiceti minut nepřihlásil do systému, uvítá ho hlavní nabídka která je na Obr. 10.



Obr. 10: **Úvodní stránka (nepřihlášený uživatel)**

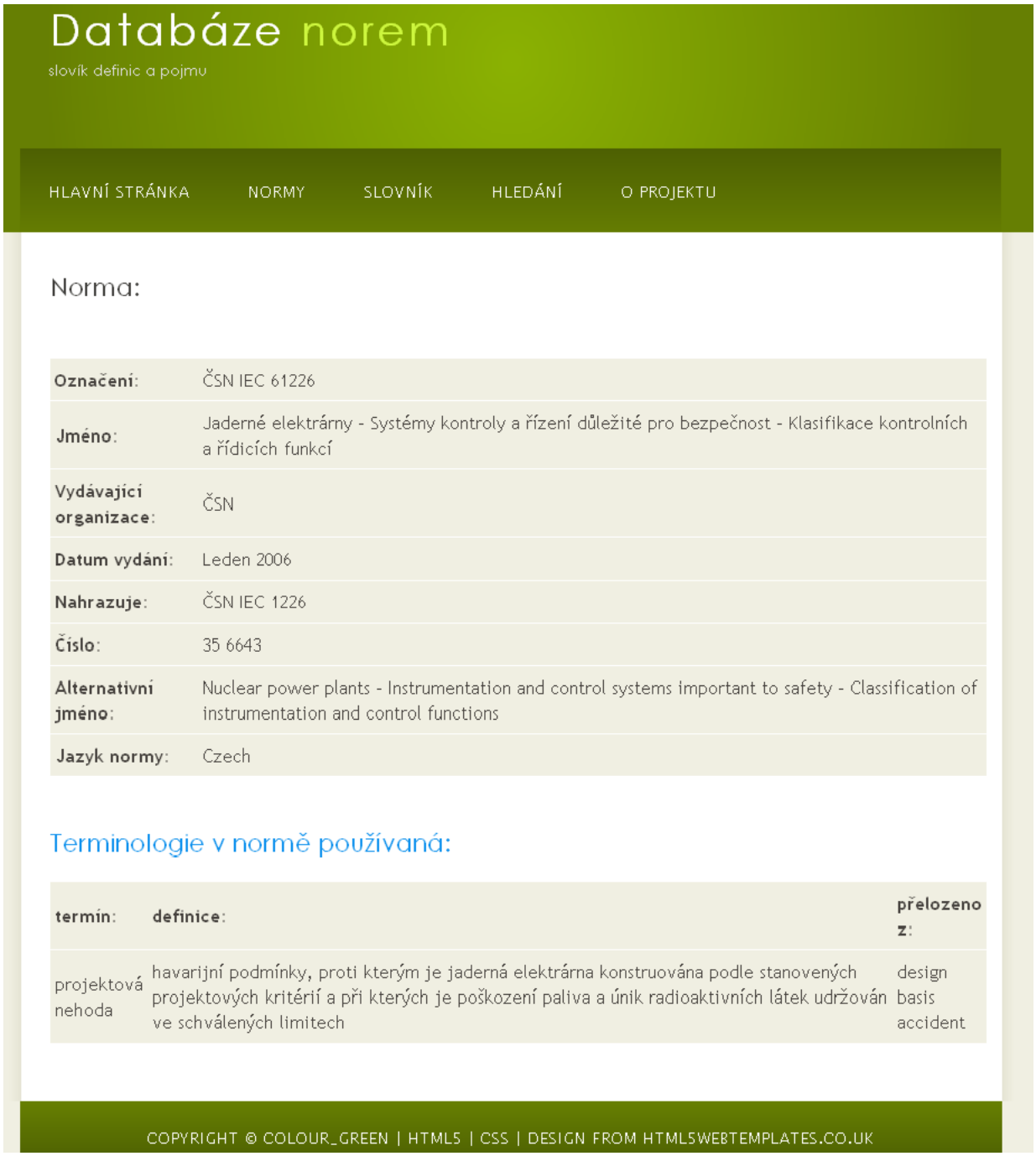
Je k dispozici několik následující možností: Přihlásit se (pokud nějakou registraci má)

Zaregistrovat se (pokud ještě registraci nemá)

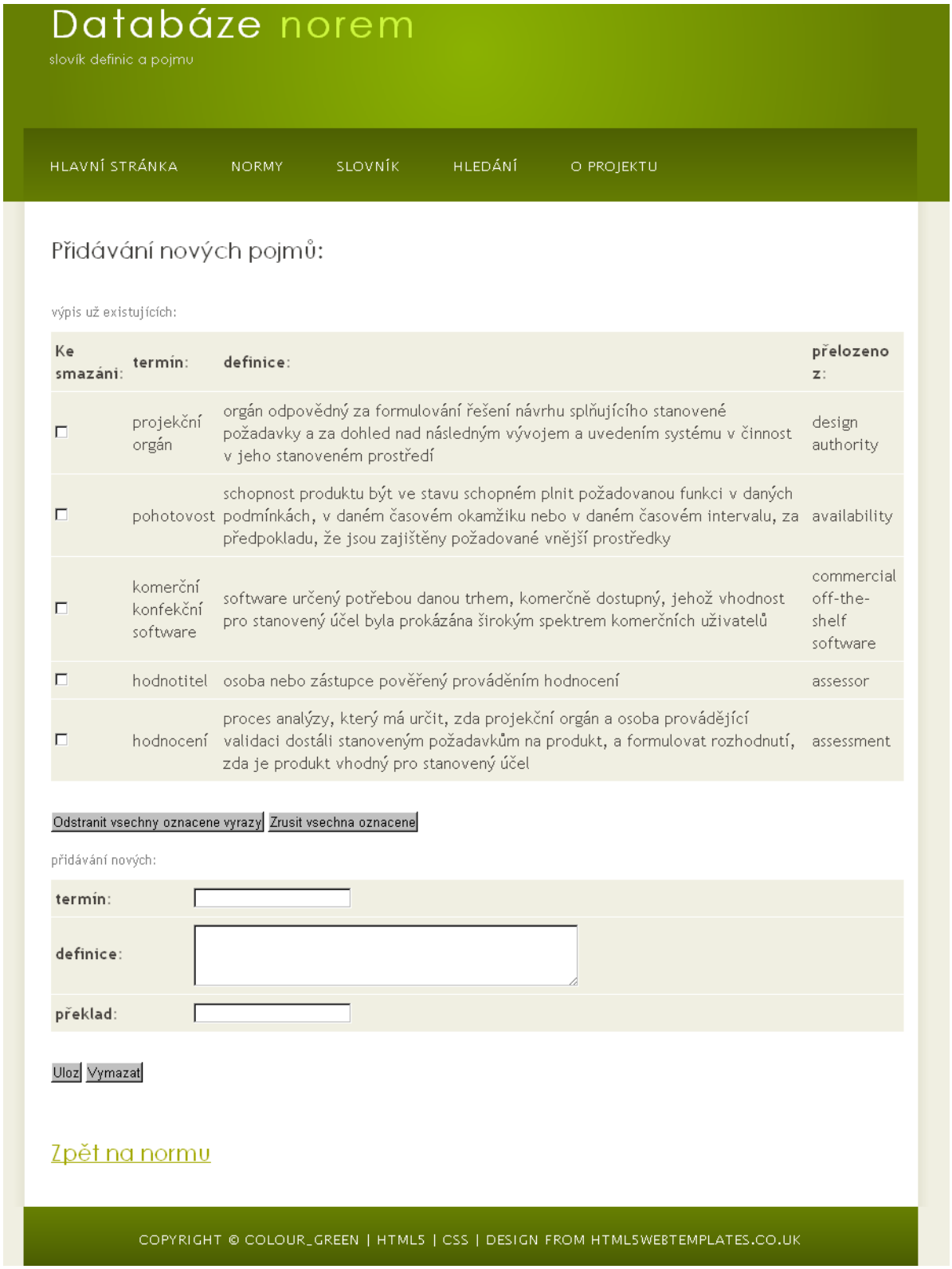
Dále i anonymní uživatelé si ale mohou prohlédnout jaké normy a termíny jsou v databázi nahrané pomocí veřejného slovníku a veřejné databáze norem, také je možné nějakou normu

vyhledat podle názvu. Nepřihlášení uživatelé nemají žádnou možnost, jak do databáze cokoli přidat nebo něco upravit.

Kromě těchto několika možností mají přihlášení uživatelé k dispozici několik dalších funkcionalit. Jedná se především o přidávání dalších norem, dále mohou ke svým normám přidávat a editovat terminologii v nich použitou (viz Obr. 11 a Obr. 12).



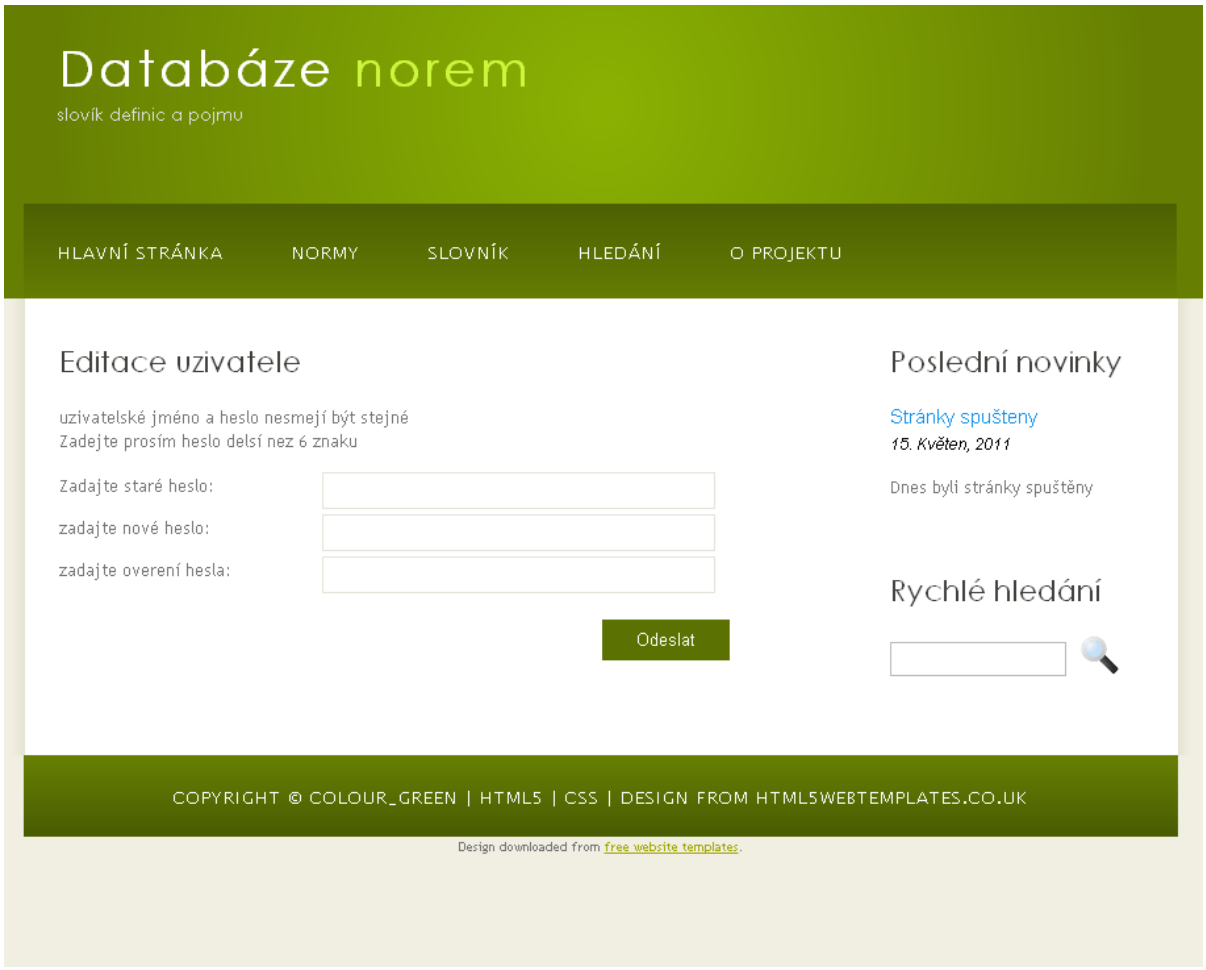
Obr. 11: **Náhled na normu**



Obr. 12: **Editace terminologie (přidávání, mazání)**

Dále si uživatelé mohou prohlédnout seznamy norem, které přidali a také slovník termínů, který vznikl z jejich norem. Uživatel může svoji normu z databáze zcela vymazat. Je rovněž možné změnit staré heslo za nové (viz Obr. 13 a Obr. 14).

Obr. 13: **Výpis norem, které přidal uživatel**

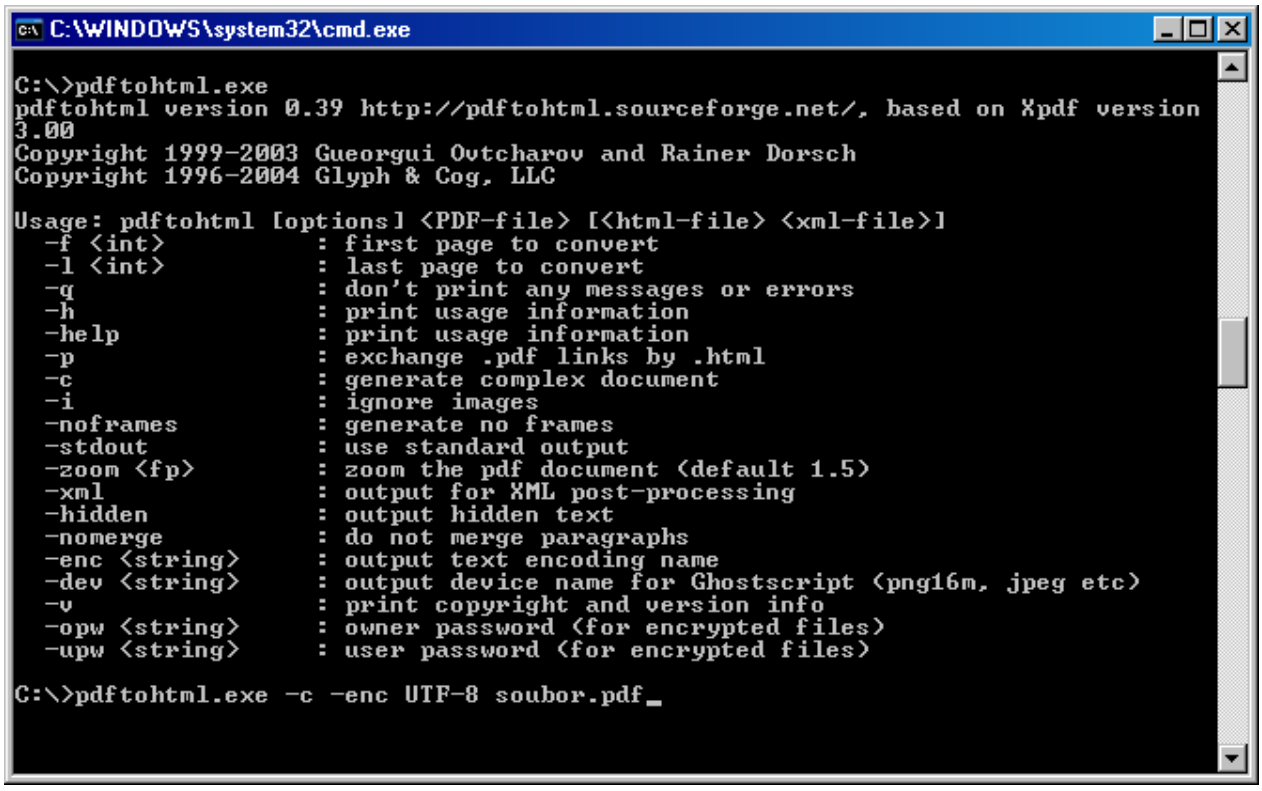


Obr. 14: **Změna uživatelského nastavení**

8 Parsování textu

V počátcích tvorby aplikace se pracovalo s myšlenkou, kdy by kromě ručního zadávání údajů do databáze obsahovala také vstup automatizovaný. Elektronická distribuce norem probíhá většinou ve formátu PDF (Portable Document Format). Pokud by tedy uživatel vlastnil PDF nějaké normy, mohl by zpracování údajů přenechat aplikaci. Ta by pomocí speciálního programu vyseparovala z normy všechny žádané informace a sama je do databáze přidala.

Autor práce měl k dispozici několik norem staršího data. Na nich provedl přezkoumání proveditelnosti parsování souboru PDF na obyčejný text. To se podařilo pomocí nástroje PDFTOHTML [14].



Obr. 15: **Rozhraní nástroje PDFTOHTML**

PDFTOHTML se ovládá z příkazového řádku a lze nastavit různé způsoby převodu. Pokud jsou navíc dodány knihovny z balíku AFPL Ghostscript, je možné převádět nejen text, ale celé soubory PDF včetně grafiky. Poté by stačilo naprogramovat v jazyce Java aplikaci založenou na regulárních výrazech, která by kýžené textové řetězce ze souborů vyseparovala.

Ukázalo se ovšem, že v dnešní době jsou normy šířeny v zašifrované podobě, kdy je lze otevřít pouze v Adobe Readeru, který má nainstalovaný zásuvný modul ochrany proti šíření. Takové soubory ovšem nástroj PDFTOHTML převést neumí, a proto byla celá idea automatického vstupu zavržena.

9 Závěr

Cílem této diplomové práce bylo vytvoření databáze norem, pomocí které bude možné u jednotlivých norem pracovat s jejich použitou terminologii a dále tuto terminologii i samostatně prohlížet. První bod zadání dává za úkol seznámit se s formátem dokumentů norem, což popisuje první kapitola práce. Je zde vysvětleno, co to normy jsou, jaký je jejich účel. Dále jakým způsobem vznikají a hlavně problematika přejímání norem evropských a mezinárodních do národní soustavy norem s ohledem na nekonfliktní způsob tvoření terminologie.

Další část práce seznamuje čtenáře výběrem technologie pro realizaci praktické části práce, která obsahuje i krátké porovnání s ostatními konkurenčními technologiemi a poskytuje čtenáři stručný popis prostředí internetu.

Třetí část detailně popisuje vlastní implementaci aplikace pomocí technologie JSP, s důrazem na detailní rozbor vlastností a programovacích technik, které byli při realizaci aplikace využity.

Další část pojednává o řešení databázové části a popisuje jistá specifická řešení v práci použitá (zabezpečení uživatelských údajů a ochranu proti spamovacím robotům).

Předposlední kapitola pojednává o tom, jak aplikaci, k této práci přiloženou na CD, správným způsobem nainstalovat a zpřístupnit včetně všech podpůrných technologií (databáze, aplikační server).

V poslední části práce se čtenář dozví, jakým způsobem se aplikace správně používá, kapitola zároveň demonstruje výsledek práce.

Výsledkem práce je on-line aplikace, pomocí které si mohou různí uživatelé vytvořit svoji evidenci norem včetně terminologie v nich použité, z takto zadaných údajů pak vzniká veřejná databáze norem a terminologie v nich použité. Tím se podařilo splnit všechny body zadání.

Seznam použitých zkratek

API - Application Programming Interface

Jedná se o sbírku procedur, funkcí či tříd nějaké knihovny, které může využívat programátor, který danou knihovnu využívá.

GUI - Graphical User Interface

Grafické uživatelské rozhraní je uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků.

HTML - Hypertext Markup Language

Značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu.

HTTP- Hypertext Transfer Protocol)

Internetový protokol určený pro výměnu hypertextových dokumentů ve formátu HTML. Protokol je spolu s elektronickou poštou tím nejvíce používaným a zasloužil se o obrovský rozmach internetu.

IDE - Integrated Development Environment

Vývojové prostředí, je software usnadňující práci programátorů, většinou zaměřený na jeden konkrétní programovací jazyk.

IP - Internet Protocol

Datový protokol používaný pro přenos dat přes paketové sítě. Tvoří základní protokol dnešního Internetu.

JDBC - Java Database Connectivity

API pro programátory v programovacím jazyku Java, které definuje jednotné rozhraní pro přístup k relačním databázím. Pro přístup ke konkrétnímu databázovému serveru je potřeba JDBC driver (ovladač), který poskytuje tvůrce databázového serveru.

JDK - Java Development Kit

Soubor základních nástrojů pro vývoj aplikací pro platformu Java.

JRE - Java Runtime Environment

JVM a API společně tvoří celek, který je poskytován jako Java Runtime Environment.

JSP - JavaServer Pages (JSP)

Java technologie, která pomáhá softwarových vývojářům ve vývoji dynamicky generovaných webových stránek založených na HTML nebo XML.

JVM - Java Virtual Machine

Java Virtual Machine je sada počítačových programů a datových struktur, která využívá modul virtuálního stroje ke spuštění dalších počítačových programů a skriptů vytvořených v jazyce Java.

SQL - Structured Query Language

Standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.

TCP - Transmission Control Protocol

Protokol internetu představující transportní vrstvu. Protokol garantuje, že paket bude spolehlivě doručen a navíc ve správném pořadí doručování ve správném pořadí.

UNMZ - Úřad pro technickou normalizaci, metrologii a státní zkušebnictví

Úřad zabezpečení výkonu státní správy v oblasti technické normalizace, metrologie a státního zkušebnictví.

WWW - World Wide Web

Často taká nazýváno jednoduše jako Web. Je soustava propojených hypertextových dokumentů.

Literatura

[1] Java Platform Standard Edition 6 API Specification
URL: < http://java.sun.com/javase/6/docs/api/index.html>
[citováno 2. května 2011]

[2] Wikipedia
URL: < http://www.wikipedia.org > [citováno 6. května 2011]

[3] BSD
URL: < http://www.opensource.org/licenses/bsd-license.php >
[citováno 8. května 2011]

[4] Zakhour, Sharon. Java 6 - Výukový kurz. 1. vydání Brno: Computer Press, 2007. ISBN 978-80-251-1575-6

[5] GNU General Public License
URL: < http://www.gnu.org/licenses/gpl.html> [citováno 8. května 2011]

[6] Barry Burd JSP: Java Server Pages podrobný průvodce. Computer Press, a.s., 2003. Vydání první. ISBN: 807226804X. 416 str.

[7] Úřad pro technickou normalizaci, metrologii a státní zkušebnictví
URL: < http://www.unmz.cz > [citováno 8. května 2011]

[8] Vlastní TAGY v JSP
URL: < http://www.root.cz/clanky/vlastni-lttagy-v-jsp/ >
[citováno 12. května 2011]

[9] MySQL
URL: < http://www.mysql.com/ > [citováno 12. května 2011]

[10] IEC Glossary
URL: < http://std.iec.ch/glossary > [citováno 14. května 2011]

[11] IEC 60050 – International Electrotechnical Vocabulary
URL: < http://www.electropedia.org/ > [citováno 14. května 2011]

[12] Java SE Downloads
URL:< http://www.oracle.com/technetwork/
java/javase/downloads/index.html/ > [citováno 15. května 2011]

[13] Apache Tomcat
URL: < http://tomcat.apache.org/ > [citováno 15. května 2011]

[14] PDFTOHTML
URL: < http://pdftohtml.sourceforge.net/ > [citováno 15. května 2011]

[14] BOZP
URL: < http://www.bozpinfo.cz/win/knihovna-bozp/citarna/clanky/
technicka_bezpecnost/techn030428.html > [citováno 15. května 2011]

[15] PHP Guru
URL: < http://www.phpguru.cz/clanky/soleni-hesel > [citováno 15. května 2011]

[16] Jiří Peterka: Captcha
URL: < http://www.earchiv.cz/b06/b0912001.php3 > [citováno 15. května 2011]

[17] Výukové materiály
URL: < http://www.osuchowski.cz/wwwstranky/mysql.php >
[citováno 15. května 2011]

[18] Úvod do JDBC
URL: < http://interval.cz/clanky/uvod-do-jdbc/> [citováno 16. května 2011]

[19] Prodejna norem ČSN
URL: < http://shop.normy.biz/> [citováno 16. května 2011]

[20] Metrologická terminologie
URL: < http://www.sekk.cz/terminologie/Text/Terminologie.htm>
[citováno 16. května 2011]

[21] Technické normy ČSN
URL: < http://www.technicke-normy-csn.cz/normy-csn-pojem-tvorba.html>
[citováno 16. května 2011]