Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Bakalářská práce

Využití objektového přístupu v internetových aplikacích

Autor: Adam Sádovský
Vedoucí práce: Doc. Ing. Vojtěch Merunka, PhD.

Čestné prohlášení

Prohlašuji, že jsem tuto bakalářskou práci na téma „Využití objektového přístupu v internetových aplikacích“ vypracoval samostatně za použití uvedených zdrojů, svých poznatků a konzultací s vedoucím práce**.**

Adam Sádovský

Poděkování

Tímto bych rád poděkoval panu Doc. Ing. Vojtěchu Merunkovi, PhD. za jeho čas, cenné připomínky a odborné vedení při zpracování této práce.

Využití objektového přístupu v internetových aplikacích

Souhrn

Obsahem práce je popis, porovnání a demonstrace objektové technologie související s vývojem webových aplikací. Jsou přiblíženy současné objektové jazyky a nástroje, které se k vývoji využívají a podrobněji je popsán framework Seaside, ve kterém také byla implementována ukázková aplikace.

Klíčová slova

Objektové jazyky, internetové aplikace, webový framework, Java, PHP, Python, Smalltalk, Seaside

Usage of object technologies in internet applications

Summary

This work describes, compares and demonstrates object technologies used in web applications development. It describes object languages and development tools and mainly focus on Seaside web framework, that was used to implements demonstration application.

Keywords

Object-oriented languages, internet applications, web framework, Java, PHP, Python, Smalltalk, Seaside

Obsah

1Úvod....3
2Cíl a metodika....4
3Jazyky....5
3.1Představení objektových jazyků....5
3.1.1Stručný popis....5
3.1.2PHP....6
3.1.3Java....7
3.1.4Smalltalk....8
3.1.4.1VisualWorks....9
3.1.4.2Squeak....9
3.1.5Python....9
3.2Porovnání PHP, Java, Smalltalk a Python....10
4Webové servery....10
4.1Jazyky PHP, Java, Smalltalk a Python na straně serveru....10
4.1.1PHP....11
4.1.2Java....11
4.1.3Python....11
4.1.4Smalltalk....12
4.2Apache....13
4.3Internet Information Services....13
4.4AOLserver....13
4.5Souhrn....14
5Frameworky....16
5.1Porovnání....18
5.2Vývoj....18
5.3Model-View-Controller....18
6Seaside....19
6.1Technologie....19
6.1.1Linearita programování....19
6.1.2Hierarchie komponent....20
6.1.3Backtracig....20
6.1.4Dekorátory....21
6.2Klady....21
6.3Zápory....21
7Realizace jednoduchého
účetního systému s webovým přístupem....22
7.1Analýza uživatelských potřeb....22
7.2Návrh systému....24
7.2.1Objektový model....24
7.2.2Realizace přístupu k modelu....25
7.3Implementace části systému....26
7.3.1Model....26
7.3.2Web....27
8Závěr....32
9Seznam literatury....33

1 Úvod

Počátky webu lze vystopovat v sítích spojujících sálové počítače s terminály. Toto řešení nabízelo centralizovaný výkon, jednotný přístup k bázím dat a oddělené uživatelské účty.

Zmíněný přístup byl potlačen příchodem PC a rozměrných aplikací zajišťujících komplexní přístup k datům. Jejich problémem ovšem byla nekompatibilita a obtížná rozšiřitelnost.

S rozvojem internetu se původní myšlenky vrací. Místo sálových počítačů ovšem nastupují sofistikované firemní sítě a místo terminálů webový klienti. Jako standardní formát přenosu dat na internetu je považováno HTML a jeho pokračovatelé (především XHMTL) a od okamžiku kdy HTML podporuje formuláře, slouží zároveň jako standardizované uživatelské rozhraní.

Vznikly dynamicky generované stránky prezentující centrálně spravovaná data a nabízející uživateli jisté možnosti jejich administrace. Tento systém se ukázal vhodný především pro masový přístup ke službám, které firmy nabízely běžným zákazníkům, u kterých se nedala předpokládat existence specializované aplikace.

Problémem webu ovšem zůstává protokol HTTP, který je bezstavový a tudíž v důsledku nerozpoznává uživatele a jeho jednotlivé přístupy. Ale není to jediný problém, který musí vývojáři internetových aplikací překonávat. Požadavky kladené klienty na funkčnost jak informačních systémů tak i webových aplikací, se rychle mění. Navíc také roste komplexnost informačních systémů. Jako jedno z nejlepších řešení se ukázalo objektově orientované programování. [1]

Objevily se tak i řešení internetových aplikací založené na objektovém přístupu. Nemusí jít vždy o webové aplikace (aplikace komunikují s uživatelem přes protokol HTTP), ale právě ty jsou dnes nejpopulárnější a nejužívanější.

2 Cíl a metodika

Cílem je shrnout a porovnat objektové jazyky, nástroje a technologie využívané k vývoji internetových aplikací. Jde tedy především o popis a porovnání reprezentativního výběru objektových jazyků, dále webových serverů a konečně webových frameworků.

První část práce je především popisem současného stavu na scéně objektových jazyků. Postupně popíše jednotlivé jazyky a také vztah těchto jazyků k webovým aplikacím k jejich pozici na straně serveru. Nakonec porovná tyto jazyky vzhledem k vybraným vlastnostem.

Druhá část práce se zaměří na konkrétní vývojový nástroj – tedy na framework Seaside postavený na jazyce Smalltalk. Vyzdvihne jeho kladné i záporné stránky a nakonec uvede demonstrační příklad jednoduché internetové aplikace postavené právě na této moderní technologii.

3 Jazyky

3.1 Představení objektových jazyků

Objekty jsou v programovacích jazycích téměř prestižní záležitostí a pokud ten či onen programovací jazyk neumí „objekty“, tak vlastně ani není. Jazyků, které umí pracovat s objekty, je mnoho, ale těch které jsou opravdu objektové, nebo alespoň implementují větší část vlastností vyplývajících o teorie objektů je už poskrovnu.

Jazyky, které mají co do činění s objekty lze dělit například takto:

- Čistě objektové: **Smalltalk**, **Eiffel**, **Ruby**
- Objektové s procedurálními prvky: **Java**, **Python**
- Procedurální jazyky rozšířené o objekty (třídy): **C++**, **Fortran 2003**, **Perl**, **PHP**

Podle: [2]

Z těchto zmíněných se některé více a jiné méně hodí pro programování webových aplikací. Následující stránky se budou zabývat těmi objektovými jazyky, které se užívají nejčastěji: **PHP**, **Java**, **Smalltalk**, **Python.**

3.1.1 Stručný popis

PHP je nejpoužívanější jazyk pro generování webu. Jeho kladem je snadná pochopitelnost, rozsáhlá podpora pro nejrůznější databáze, stejně jako dostupnost mnoha knihoven a hotových skriptů. Objekty plnohodnotně podporuje od verze 5.0.

Java se stává dominantním jazykem v rozsáhlých firemních aplikacích. Rychlý vývoj posledních let zaručuje dostupnost technologie a v neposlední řadě také vývojářů. Podpora objektového přístupu je velmi dobrá.

Smalltalk a především webový framework Seaside je velmi vhodný pro rychlé budování webových aplikací. Při generování webu pracuje na vyšší úrovni abstrakce a zakrývá před programátorem nedostatky protokolu HTTP. Podpora objektů je výborná.

Python je oblíbený především díky podpoře jiných programovacích jazyků. Hotové aplikace se díky skriptu v Pythonu mohou snadno modifikovat. Podpora objektů je dobrá.

3.1.2 PHP

je od roku 1994 nejpoužívanější skriptovací jazyk pro generovaní webových stránek. Vytvořil ho z jazyka **Perl Rasmus Lerdorf** pro vlastní potřebu. Opodstatněná popularita PHP staví na snadno pochopitelné syntaxi, schopnostem jazyka, ke kterým patří i objektový přístup a v neposlední řadě i na rozsáhlých knihovnách. PHP podporuje velké množství databázových serveru MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL a SQLite. S verzí 5 se objektový přístup v PHP výrazně zlepšil a je podobný jazyku Java. [3]

Implementace:

PHP je skriptovací jazyk a je spustitelný na jakékoli platformě, pokud pro ní existuje PHP interpreter. Běžná je kombinace PHP a serveru Apache, který ho interpretuje, ale existují i verze, které dovolují v tomto jazyce vytvářet plnohodnotné aplikace:

- **PHP-GTK** – je rozšíření PHP o implementaci GTK knihoven (http://gtk.php.net/)
- **WinBinder** – rozšíření, které PHP dovoluje vyvíjet aplikace pro systém Windows (http://www.hypervisual.com/winbinder/)
- **PHP-Qt** – rozšíření dovolující PHP pracovat s knihovnami Qt (http://php-qt.berlios.de/)
- **Klorofil** – je open source iniciativa snažící se o rozšíření PHP do oblasti vývoje podnikového softwaru. Nejdůležitější součástí Klorofilu je gambArt pro vývoj okenních aplikací (http://www.klorofil.org/)

[4, 5]

3.1.3 Java

Zrod jazyku Java leží na počátku 90 let, kdy byl ve firmě **Sun Microsystems** započat projekt objektově orientovaného jazyka podobného C++, nezávislého na platformě. Java ovšem není čistě objektová, nýbrž obsahuje i přístupy „klasického“ programování jako jsou primitivní typy a statické metody, jež jsou obdobou funkcí například z C/C++.

Java podobně jako Smalltalk vyžaduje ke svému běhu virtuální stroj, ale důvodem, proč je Java interpretovaná, je spíše požadavek nezávislosti na platformě, než nutnost překlenování nedostatečných schopností hardwaru. Virtuální stroj Javy je v současnosti využíván i jinými jazyky, které po přeložení do bytecodu běží na takzvané Java platformě. [6]

Implementace:

Vývoj jazyka je v současnosti koordinován JCP (Java Community Process) zatímco implementace je téměř výlučně doménou společnosti Sun. Od ní přichází rozmanité množství distribucí nejrůznějších verzí a zaměření:

- **Java EE** (předtím J2EE) (Java Platform, Enterprise Edition) – pro distribuované podnikové aplikace
- **Java ME** (předtím J2ME) (Java Platform, Micro Edition) – pro PDA a mobilní telefony
- **JDBC** (Java Database Connectivity) – pro nezávislé připojení k databázím
- **JSP** (JavaServer Pages) – pro generování webových stránek
- **Java 3D** – pro 3D programování
- a další.

Jiné distribuce jsou velmi minoritní a jde například o **Blackdown** (http://www.blackdown.org/) Java for Linux nebo GNU **Classpath** (http://www.gnu.org/software/classpath/)

Podle: [7]

3.1.4 Smalltalk

Smalltalk se nechal inspirovat jazyky jako **Simula**, **Sketchpad** a **Lisp**, je čistě objektový, což značí, že do důsledku respektuje teorii objektů a je vlastně její aplikací. Původní koncepce je stará téměř 35 let, přesto i dnes ukazuje směr vývoje objektů. Vývoj započala společnost **Xerox** v čele s **Alanem Kayem**, **Tedem Kaehlerem**, **Adele Goldbergovou** a **Danem Ingalls**. Cílem Smalltalku byla a je symbióza lidí a počítačů, snadno pochopitelná koncepce a ovládání přes grafické rozhraní. Samotný kód Smalltalku respektuje lidsky čitelný zápis.

Na současném hardwaru běží Smalltalk jako interpretovaný a vyžaduje pro svůj běh takzvaný virtuální stroj, který překlenuje nedostatky současných počítačů. K interpretaci dochází na úrovni bytecodu, pomocí JIT compileru, nejde tedy o interpretaci přímo zapsaného zdrojového kódu, ale kódu přeloženého v okamžiku potřeby pro virtuální stroj. [8, 9]

Implementace:

Smalltalk jako koncepce je k dipozici v několika implementacích:

- **F-Script**
- **IMB VisualAge**
- **S#**
- **Squeak**
- **VisualWorks**
- a další

Podle [8]

3.1.4.1 VisualWorks

je komerční implementací společnosti **Cincom** vycházející přímo ze Smalltalku-80, který později nesl název ObjectWorks a následně VisualWorks. Jde o zásadní implementaci Smalltalku s množstvím vyspělých funkcí a vlastností. Nekomerční verze je společností nabízena zdarma. [10]

3.1.4.2 Squeak

také vychází z původního Smalltalku-80, ale jde o open source projekt a je k dispozici zdarma i se zdrojovými kódy. Významným spolupracovníkem na něm je **Alan Kay**, který se podílel na vývoji Smalltalku a Dynabooku v jejich zrodu. Uživatelské grafické rozhraní (GUI) simuluje typicky systém Smalltalku uvnitř jediného okna, ale existují i rozhraní založené na wxWidgets. [11, 12]

3.1.5 Python

Jde o jazyk hybridní, ve kterém je možné programovat jak objektově, tak i procedurálně. Původní koncepce vychází z požadavku jednoduchosti syntaxe, jako i čistoty a čitelnosti kódu. Byl navržen v roce 1990 **Guido van Rossumemem** a v současnosti je vyvíjen jako open source.

Vlastností Pythonu je i spolupráce s jinými jazyky a to jako jazyka skriptovacího, pomocí kterého lze snadno a pružně rozšiřovat funkčnost již hotové aplikace.

Implementace:

Python je k dispozici pro velké množství platforem a ve velkém množství jazyků:

- **Jython** - implementace Pythonu napsaná v Javě
- **IronPython** - implementace Pythonu napsaná v C# a určená pro .NET a Mono
- **PyPy** – implementace Pythonu napsaná v Pythonu

Podle: [13]

3.2 Porovnání PHP, Java, Smalltalk a Python

Porovnávací tabulka je čistě orientační a většinou odráží názor autora. Některé informace jsou ovšem získané z: [5, 7, 8, 13]

	PHP	Java	Smalltalk	Python
Třídy	●●	●●●	●●●	●●
Dědění	●●	●●●	●●●	
Rozhraní	●●●	●●●	●	
Skládání (agregace, kompozice)	●●●	●●●	●●●	●●●
Polymorfismus	●●	●●	●●●	
Zapouzdření	●●	●●	●●●	●
Aktorový model (delegování)	●	●	●●	
Přetěžování metod	●●	●●●	●	●
Přetěžování operátorů	●	●	●●	●●●
Metatřídy (přístup k třídním metodám)	●	●●	●●●	
Paralelismus		●●●	●●●	
Klient/server	●	●●●	●●●	
Garbage Collection	●	●●●	●●●	●●●
Typová kontrola při překladu	●	●●●	●	
Typová kontrola za běhu	●●	●	●●●	
Přístupová práva	●●	●●●	●●	
Integrace s jinými jazyky	●●	●●	●	●●●
Ø	**1,75**	**2,41**	**2,35**	**2,29**

Tabulka 1: Porovnání vybraných jazyků;
Počet bodů ukazuje jak propracovaná je podpora hodnocené vlastnosti v tom kterém jazyku.

4 Webové servery

4.1 Jazyky PHP, Java, Smalltalk a Python na straně serveru

Všechny zmíněné jazyky umí komunikovat s „okolím“ přes síť pomocí nejrůznějších protokolů jako HTTP, FTP, SQL, … s využitím mnoha přístupů typu klient/server, peer-to-peer a tak dál. K této komunikaci využívají specializovaných knihoven, ale existují i rozměrné aplikační frameworky připravené pro vývoj systému například pro webová rozhraní.

Tak mohou vzniknout servery napsané přímo v jednom ze zmíněných jazyků. Nadto existuje i mnoho modulů nebo „mostů“ rozšiřujících jazykové schopnosti téměř

jakéhokoli serveru (ať už je napsán v C++ nebo třeba v Pythonu). Jde-li konkrétně o generování webových stránek, jsou všechny možné kombinace téměř nevyčerpatelné.

4.1.1 PHP

PHP je typicky skriptovací jazyk a vývoj aplikací se v něm nepředpokládá, přesto existují projekty, které rozšiřují schopnosti jazyka i o tuto možnost. Hlavní předností PHP je snadná pochopitelnost, podobnost s C++ a úzký vztah se serverem **Apache**. PHP ve spojení se zmíněným serverem Apache je nejpoužívanějším webovým řešením vůbec.

4.1.2 Java

Java je v současné době tak rozlehlou platformou, že je na ní možné vyvíjet téměř jakýkoli software, nebo aplikaci. Existují například tyto implementace webového serveru v jazyce Java:

- **Jigsaw** - W3C's Server – implementující HTTP 1.1 (http://www.w3.org/Jigsaw/)
- **Jetty** – je HTTP server a Servlet Container (http://jetty.mortbay.org/jetty/index.html)
- **Blazix** – webový server (http://www.blazix.com/)
- **Sun Java System Web Server** – web server přímo od firmy Sun implementuje technologie JSP, PHP, CGI a další

  (http://www.sun.com/software/products/web_srvr/home_web_srvr.xml)

  Podle: [14]

4.1.3 Python

Python přestože je především skriptovací jazyk, umožňuje plnohodnotný vývoj širokého spektra aplikací (a to i webových serverů). Seznam většiny projektů toho ražení je zde: http://wiki.python.org/moin/WebProgramming.

- **Twisted** – je rozsáhlý projekt implementující velké množství komunikačních protokolů jako TCP, UDP, multicast, SSL/TLS, serial communication. Jeho součástí je i webový server. Vyvíjen je v Pythonu pod licencí MIT.

  (http://twistedmatrix.com/trac)

- **Pylons** – je moderní framework pro vývoj webu (http://pylonshq.com/)

- **PythonWeb** – nabízí moduly nutné pro vývoj webových aplikací

  (http://www.pythonweb.org/)

- **Snakelets** – je velmi snadno použitelný webový server

  (http://snakelets.sourceforge.net/)

- **Mod_python** – je modul pro široce rozšířený webový server Apache, připojující k severu interpreter Pythonu (http://www.modpython.org/)

  Podle: [14]

4.1.4 Smalltalk

Pro **Smalltalk** je k dispozici velké množství balíků rozšiřujících schopnosti konkrétní implementace. Tyto balíky jsou ekvivalentem programů napsaných pro nějaký operační systém, s tím, že s sebou nesou všechny výhody objektů.

- **Comanche** – je HTTP server dostupný v balíku KomHttpServer-6.1.sar (poslední stabilní verze) (http://squeaklab.org/comanche/)

- **Seaside** – framework pro vývoj webových aplikací (http://www.seaside.st/)

- **Croquete** – je projekt spojující software se síťovou komunikací zajišťující spolupráci velkého množství lidí a sdílení zdrojů (http://www.opencroquet.org)

  Podle: [12]

4.2 Apache

Jedním z nejpoužívanějších HTTP serverů je pod open-source vyvíjeny server Apache. Je k dispozici zdarma pro velké množství platforem (UNIX, Windows). Splňuje řadu požadavků na bezpečnost, konfigurovatelnost i rozšiřitelnost. Je velmi rychlý a díky stále stoupající oblibě jsou pro něj v současné době k dispozici balíky rozšiřující jeho funkčnost. Na stránkách Apache Software Foundation (http://www.apache.org/) zajišťujících podporu komunitě Apache open-source software projects, je seznam projektů, které tento server využívají.

V souvislosti s jazyky PHP, Java a Python jsou zajímavé tyto projekty:

- **Jakarta Project** – organizující projekty postavené na jazyce Java (http://jakarta.apache.org/)
- **Apache Tomcat** – implementující technologii Java Servlet a JavaServer Pages do HTTP serveru Apache (http://tomcat.apache.org/)
- **PHP** – jazyk PHP je ze své podstaty spojen s HTTP serverem Apache. Je dostupný jako modul nebo jako CGI skript (http://www.php.net/)
- **Mod_python** – je modul Apache, připojující k severu interpreter Pythonu (http://www.modpython.org/)

Podle: [15]

4.3 Internet Information Services

IIS je v pořadí další nejpoužívanější webový server od firmy Microsoft. Zvládá protokoly jako FTP, SMTP, NNTP a HTTP/HTTPS. Hlavním skriptovacím jazykem je ASP, ale podobně jako Apache je možné rozšíření o jazyky Perl, Python, Tcl... (http://www.activestate.com/) [16]

4.4 AOLserver

Je open-source web server podporující Tcl a C. Rozšiřitelnost je ale oproti serveru Apache a IIS minimální, už jen ze strany menšího zájmu vývojářů.

4.5 Souhrn

Nemusí nutně platit, že rozsáhlé a všeobjímající komerční řešení webového serveru je nejlepší. Jak při volbě platformy (Windows versus GNU Linux), tak i softwaru (Apache versus IIS) se silně prosazují nekomerční projekty s otevřeným zdrojovým kódem. Tato skutečnost – tedy otevřenost zdrojového kódu – má dopad na množství a škálovatelnost softwarových řešení webových aplikací. Za prvé je z čeho vybírat a za druhé lze spojovat ty prvky, jež ideálně splňují požadavky na výslednou funkčnost. I uzavřená komerční řešení musela přistoupit na určitou modulárnost. Proto na serveru IIS od firmy Microsoft, není problém využívat výhod jazyka Python.

	PHP	Python	Tcl	ASP	JSP	Smalltalk
Apache	Modul	Modul	Ano	Modul	Ne	Ne
IIS	Ne	Ne	Ne	Ano	Ne	Ne
AOLserver	Ne	Ne	Ano	Ne	Ne	Ne
Comanche	Ne	Ne	Ne	Ne	Ne	Ano
Sun Java System Web Server	Ano	Ne	Ne	Ano	Ano	Ne
Jetty	Ne	Ne	Ne	Ne	Ano	Ne
Apache Tomcat	Ne	Ne	Ne	Ne	Ano	Ne
Twisted	Ne	Ano	Ne	Ne	Ne	Ne

Tabulka 2: Implementace jazyků ve webových serverech

	Http	Https	CGI	Servlet	Fast CGI	SSI
Apache	Ano	Ano	Ano	Ne	Ano	Ano
IIS	Ano	Ano	Ano	Ne	Ne	Ano
AOLserver	Ano	?	?	?	?	?
Comanche	Ano	Ano	Ne	Ne	Ne	?
Sun Java System Web Server	Ano	Ano	Ano	Ano	Ano	Ano
Jetty	Ano	?	Ano	Ano	?	?
Apache Tomcat	Ano	Ano	Ano	Ano	Ne	Ano
Twisted	?	?	?	?	?	?

Tabulka 3: Implementace technologií ve webových serverech

	Windows	Mac OS X	Linux	BSD	Solaris
Apache	Ano	Ano	Ano	Ano	Ano
IIS	Ano	Ne	Ne	Ne	Ne
AOLserver	Ne	?	Ano	Ano	Ano
Comanche	Jako VM	?	Jako VM	?	?
Sun Java System Web Server	Ano	Ne	Ano	Ne	Ano
Jetty	Ano	Ne	Ano	Ne	Ano
Apache Tomcat	Ano	Ano	Ano	Ano	Ano
Twisted	Ano	?	Ano	?	?

Tabulka 4: Dostupnost webových serverů na platformách

Podle: [14]

To, že u konkrétního serveru není vypsán nějaký skriptovací jazyk, neznamená, že neexistuje možnost jak tuto schopnost k serveru dodatečně přidat. Také je v tomto místě asi vhodné poznamenat, že jazyk ASP není ani tak samotný jazyk, jako spíš přístup jak přímo do (X)HTML kódu vkládat skripty, které se provedou na straně serveru, ještě před odesláním straně klienta (běžně to bývají skripty v jazyku VBScript, JScript, PerlScript a další) [16].

Volba vhodného řešení bude záviset na velikosti systému, počtu modulů, počtu přístupů a objemu zpracovávaných dat. Bude též důležité jak často bude docházet ke změnám a modifikacím a jak vypadá současné řešení na které bude webový server navázán. Objektový přístup a tedy volba nějakého objektového jazyka, je vhodná zejména tam, kde je nutné rychle uzpůsobovat potřeby webových aplikací často se měnícím požadavkům, nebo tam kde se lze předpokládat rozsáhlou a komplikovanou strukturu systému.

Systém	Server	Jazyk
Linux	Apache	PHP
Windows	Apache	PHP
Windows	IIS	ASP
Windows	Sun Java systém Web Server	Java
Linux	Sun Java systém Web Server	Java

Tabulka 5: Pravděpodobně nejužívanější kombinace systému serveru a jazyka

Systém	Server	Jazyk
Linux	Comanche s frameworkem SeaSide	Smalltalk
Linux	Apache	PHP
Linux	Apache Tomcat	Java
Linux	Apache s rozšířením Mod_python	Python

Tabulka 6: Pravděpodobně nejlepší kombinace pro objektové jazyky

5 Frameworky

Anglické slovo **framework** znamená balík knihoven (tříd, objektů) a nástrojů pro vývoj aplikací na jiné, většinou abstraktnější úrovni, než jakou nabízí původní jazyk a standardní sada knihoven.

Spojení **webový framework** odkazuje na ty nástroje, které se zabývají vývojem webových aplikací. Nejčastěji nabízejí komfortnější přístup k databázím, komplexní využívání šablon a správu session.

Některé z nich jsou jednoduší a jiné složitější. Liší se také úrovní abstrakce, logikou a provedením. [17]

Existuje jistá taxonomie, kterou předkládá projekt The Web Framework Map (http://www.reahl.org/wfmwiki) a která dělí frameworky podle dvou klasifikačních kritérií. Podle view taxonomie a podle controller taxonomie.

View taxonomie má dvě hlavní skupiny. V originále: Programming Language Centric a Markup Centric. Volně přeloženo: Strategie zaměřená na programovací jazyk a strategie zaměřená na značkovací jazyk.

Controller taxonomie má čtyři hlavní skupiny. Static Files, Dynamic Content, Page Flow Centric a AJAX. Přeloženo jako: statické soubory, dynamický obsah, strategie postavená na toku stránek a AJAX (Asynchronous JavaScript Technology and XML, který využívá hlavně JavaScript na straně klienta).

Dělení je ještě jemnější, ale v zásadě ukazuje rozmanitost s jakou se webové frameworky vypořádávají s problémy vývoje internetových aplikací. [18]

Vybrané frameworky jsou například tyto:

Založené na jazyku Java:

- Struts (http://struts.apache.org/)
- WebWork (http://www.opensymphony.com/webwork/)
- Tapestry (http://tapestry.apache.org/)

Založené na jazyku PHP:

- Zend Framework (http://framework.zend.com/)
- CakePHP (http://www.cakephp.org/)
- Simphony (http://www.symfony-project.com/)
- Seagull (http://seagullproject.org/)

Založené na jazyku Python:

- CherryPy (http://www.cherrypy.org/)
- Quixote (http://www.mems-exchange.org/software/quixote/)
- Twisted (http://twistedmatrix.com/)
- WebWare (http://www.webwareforpython.org/)

Založené na jazyku Ruby:

- Ruby on Rails (http://www.rubyonrails.com/)
- Borges (http://borges.rubyforge.org/)

Založené na jazyku Smalltalk:

- Seaside (http://seaside.st/)

[19]

5.1 Porovnání

Porovnání stávajících frameworků je obtížné. Je možné je zařazovat do taxonomie podle logiky s jakou přistupují ke zpracování webových aplikací, je možné zjišťovat, které moderní techniky podporují a které ne, a také je možné hodnotit úroveň jejich abstrakce, scalabilitu nebo výkon, ale konečné hodnocení závisí především na způsobu použití.

V obrovském množství webových frameworků, které jsou k dispozici, je možné vybrat si právě takový, který bude splňovat nejlépe požadavky, jaké na něj budeme mít.

5.2 Vývoj

Přesto, jak se zdá, budoucnost bude pravděpodobně patřit vývojovým nástrojům plně odstiňujícím nedostatky protokolu HTTP a obcházející stávající logiku vývoje webových aplikací (označní Markup Centric ve výše zmíněné taxonomii). Například zavedením architektury Model-View-Controller.

5.3 Model-View-Controller

Tato architektura, vlastně návrhový vzor, vznikla v objektových jazycích jako aplikace jejich základní vlastnosti – totiž modulárnosti. Díky ní, je možné snadno a efektivně oddělit samotný informační systém (model) od způsobu jeho prezentace a práce s ním (View-Controller).

Tento návrhový vzor je nejčastěji užívaný v GUI (grafickém uživatelském rozhraní) aplikací, ale jako vzor je tak obecný, že je možné jej s úspěchem použít i při vývoji webových aplikací.

Webové frameworky, které podporují architekturu Model-View-Controller, jsou například tyto: Struts, Tapestry, CakePHP, Ruby on Rails, Borges, Seaside.

6 Seaside

Jeden z nejatraktivnějších frameworků je na Smalltalku postavený Seaside. Odstiňuje totiž většinu problémů, které vývoj webové aplikace doprovází. Jde o v úvodu zmíněný bezstavový protokol HTTP, sessions, identifikaci uživatele a tak dále. Nabízí metody programování, díky kterým se vývoj webu podobá vývoji běžného GUI (grafického uživatelského rozhraní) aplikace. [20]

6.1 Technologie

Seaside je implementovaný především v jazyce Smalltalk (je dostupný jak pro platformu Visual Works, tak i pro Squeak), ale existuje i projekt s názvem Borges portovaný ze Seaside založený na jazyce Ruby (http://borges.rubyforge.org/).

Seaside nevytváří XHTML stránky tak jak je to běžné u jazyku PHP nebo ASP či JSP. Kód se nezapisuje přímo do generovaných stránek, ale stránky jsou generovány na základě běhu aplikace.

„Běžné frameworky kopírují model protokolu HTTP. Oddělují proces tvorby HTML stránky posílané klientovi od zpracování dat zaslaných uživatelem. Poté se rozhodují, jakou stránku mají dále generovat. U tohoto přístupu je poměrně obtížné definovat složitější procesy webové aplikace. Je prakticky nemožné oddělit logiku jednotlivých komponent stránky od aplikační logiky jako takové.“ (citace z [20])

Základní odlišnosti a současně výhody Seaside jsou tyto: linearita programování, hierarchie komponent, backtracing, dekorátory a další.

6.1.1 Linearita programování

Seaside vytváří potřebnou modálnost nad protokolem HTTP. Každý odkaz, každý formulářový prvek má vlastní nezaměnitelný individuální kód, který ho identifikuje jak vzhledem k zobrazené stránce, tak vzhledem k řetězci činností, které mu předcházeli a vzhledem k uživateli (session).

Ve skutečnosti je každá session vytvořena jako samostatná instance třídy webové komponenty (to může při velkém počtu přístupů zatěžovat server, viz zápory Seaside). Kliknutí na odkaz nebo vyplnění a odeslání formuláře, nesměruje na konkrétní stránku, ale vyvolá kód programu – pošle zprávy objektům, změní atributy objektu atd...

Navíc je navržen model call: answer: který dovoluje komponentám se navzájem volat a vracet si výsledky svého běhu. [21]

6.1.2 Hierarchie komponent

Webové stránky (v případě Seaside jde o XHTML) nejsou navrhovány jako celek, ale sestávají z komponent. Jedna z komponent je v třídní metodě root označena za kořenovou. Od ní pak dále je vytvářen jednosměrný strom komponent.

Komponenty jsou vlastně instance tříd WAComponent (Web Aubergine Component), které především přetěžují metodu renderContentOn: uvnitř které probíhá samotné vykreslení komponenty.

Dobře navržená komponenta se potom může vykreslit právě tam a právě v ten okamžik, kdy je potřeba.

Na jedné webové stránce se může nacházet i více komponent. Komponenty se vzájemně mohou volat a tím se vlastně nahrazovat (jde o ekvivalent modálního okna). [20, 21]

6.1.3 Backtracig

Pod tímto názvem se skrývá schopnost Seaside pamatovat si historii všech volaných komponent. Tlačítko zpět v internetovém prohlížeči potom opravdu vrátí stav vykreslených komponent.

Zaregistrování komponenty pro backtracing je možné provést například v inicializační metodě:

```
self session registerObjectForBacktracking: self.
```

Je možné také vytvořit obalový objekt pro jednotlivé proměnné. Například:

```
promenna := WAStateHolder new.
```

A následně k této proměnné přistupovat přes metody content a content:. [20, 21]

6.1.4 Dekorátory

Dekorátory dokáží rozšiřovat schopnosti komponent a to bez zásahu do jejich kódu. Obalují původní komponenty a mohou tak rozhodovat o dalším zařazení komponent do webové aplikace. [20, 21]

6.2 Klady

Kladem je rozhodně rychlý vývoj webové aplikace pramenící z objektového přístupu a také ze způsobu jakým se XHTML stránka generuje. Programátor nekontroluje tok dat na stránce, ale vytváří poměrně kompaktní a zapouzdřené komponenty, které rozmisťuje podle potřeby.

Vysoká je také modularita tohoto řešení a znovupoužitelnost. Běh aplikace lze mnohem snadněji kontrolovat a také definovat. Kladem je také možnost měnit funkčnost aplikace za běhu. [22]

6.3 Zápory

Seaside má i svoje nevýhody. Některé z nich lze obejít snadno, některé hůře, ale musí se nimi počítat.

Jde například o problémy s vyhledávači, které k indexaci nalezených stránek potřebují stabilní internetové odkazy, které Seaside nemá.

Existují i jisté problémy se zpětnou navigací a backtracingem. Záleží na samotném vývojáři, jak dobře backtracing navrhne.

Vzhledem k tomu, že Seaside vytváří pro každou session novou instanci komponent, může při velkém počtu přístupů dojít k vysokému nárůstu paměťových nároků a ke zpomalení systému. [22]

U velkých aplikací může nastat i problém s komunikací komponent a proto se vyplatí vytvořit komplexnější systém jejich hierarchie.

7 Realizace jednoduchého účetního systému s webovým přístupem

7.1 Analýza uživatelských potřeb

Zadavatelská firma zabývající se vedením účetnictví a daňovým poradenstvím požaduje jednoduchý účetní systém s následujícími specifiky:

- neomezený počet firem
- neomezený počet uživatelů
- dynamická účetní osnova (možnost přidávat analytické účty)
- přenos konečných stavů do nového účetního období
- možnost rozlišovat účetní operace i mimo analytické účty
- možnost zpětných oprav
- výpis stavů na účtech

Dále pak z důvodu rozmístění zaměstnanců (zaměstnanci pracují doma s přístupem na internet):

- musí být systém přístupný z internetu
- k jednomu účetnictví musí mít přístup více uživatelů
- účetnictví musí být chráněné heslem
- systém musí hlásit, kdo a kdy na účetnictví pracoval a zda ještě nepracuje

Z těchto potřeby byly extrahovány:

- **funkční požadavky:**

  - přidání, odebírání a editování uživatelů, firem, účetnictví, účtů a účetních operací
  - přenosy konečných stavů účtů
  - sledování stavu účtů
  - přihlašování uživatelů a administrátora
- **ostatní (ne-funkční) požadavky:**
  - neomezený počet firem, uživatelů
  - internetový přístup

Dále pak stručná analýza datových toků, kterou znázorňuje následující matice událostí:

Data od administrátora	Data k administrátorovi	Data od uživatele	Data k uživateli
přihlášení	seznam uživatelů	odhlášení	stav účtů
odhlášení		přihlášení	seznam firem
změna šablony		editace účetnictví	seznam účetních operací
přidání uživatele		přidání účetnictví	seznam účetnictví
odebrání uživatele		odebrání účetnictví	seznam účtů
editace uživatele		editace účetní operace	
		přidání účetní operace	
		odebrání účetní operace	
		editace účtu	
		přidání účtu	
		odebrání účtu	

Tabulka 7: Matice událostí

Vztah systému k vnějším činitelům včetně příslušných datových toků, upřesňuje následující kontextový diagram:

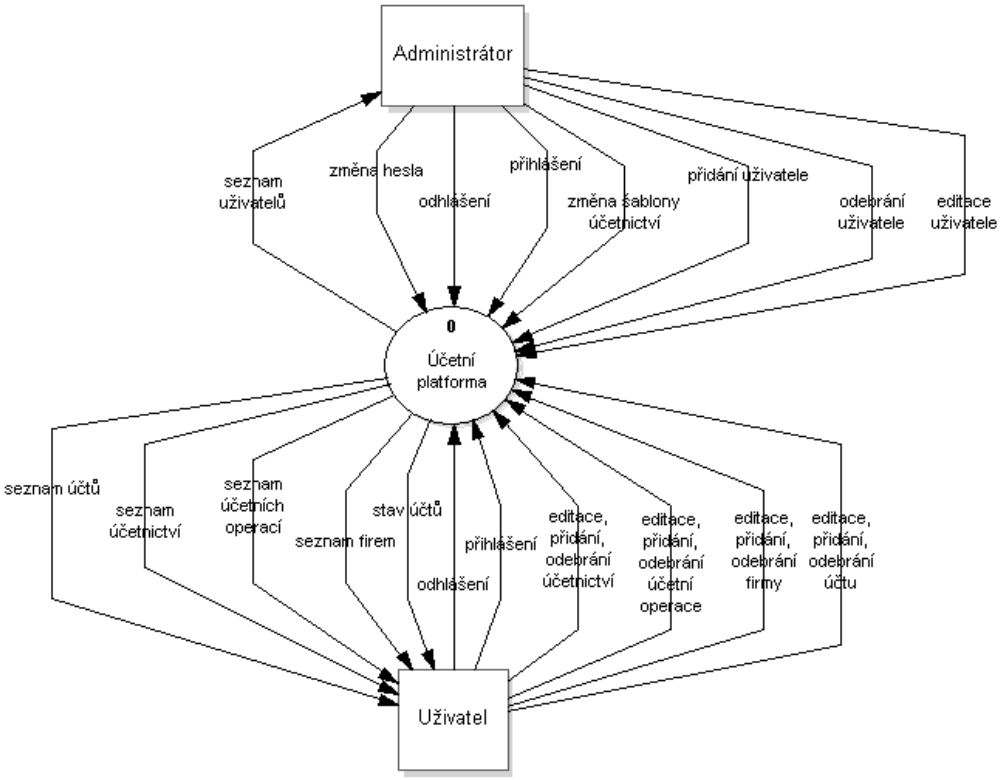


Diagram 1: Kontextový diagram účetního systému

7.2 Návrh systému

7.2.1 Objektový model

Navržený objektový model vychází z funkčních požadavků a analýzy datových toků. Představuje jádro navrhovaného systém a je funkčně nezávislý na konkrétní implementaci a na realizaci přístupu k němu. V zásadě jde o modelování podle návrhového vzoru **Model-View-Controller**.

Navrženy byly třídy Ucetnictvi, UcetniPlatforma, Uzivatel, Firma, Obdobi, UcetniOperace, Ucet, PenezniOperace, jejichž vztah a kardinalitu znázorňuje následující model tříd:

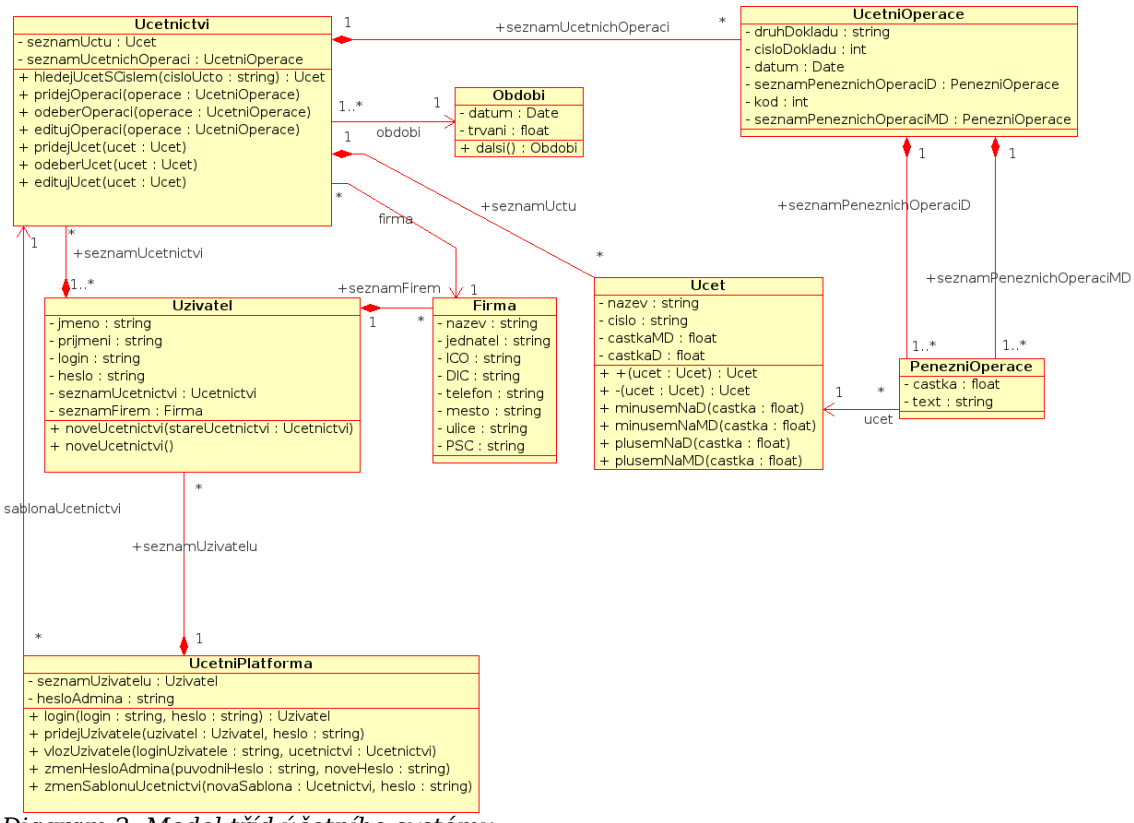


Diagram 2: Model tříd účetního systému

7.2.2 Realizace přístupu k modelu

Z analýzy ne-funkčních požadavků a z potřeby samotné bakalářské práce byl pro realizaci přístupu k účetnímu systému zvolen webový portál. Následná implementace by měla ukázat sílu objektového přístupu v případě, kdy je již navržen a odzkoušen model a kdy je potřeba vytvořit uživatelský přístup přes webové rozhraní.

7.3 Implementace části systému

Pro realizaci byl zvolen jazyk Smalltalk a to konkrétně jeho Open Source implementace Squeak ve verzi 3.8. Dále byl pro tvorbu webové aplikace, vybrán framework Seaside ve verzi 2.6.

7.3.1 Model

Model byl pro účely bakalářské práce naprogramován téměř celý.

Základem je třída **UcetniPlatforma** dovolující přidávat a odebírat uživatele, měnit heslo administrátora systému a přihlašovat uživatele do systému. Obsahuje zejména seznam uživatelů.

Příklad vytvoření instance třídy UcetniPlatforma:

```
platforma := UcetniPlatforma new.
platforma zmenHesloAdminaNa: 'vodovod' heslo: 'admin'.
adam := Uzivatel new initialize: #(Adam Sadovsky adam host).
platforma pridejUzivatele: adam heslo: 'vodovod'.
```

Atributy uvnitř objektu jsou skryté a metody, které k nim dávají přístup, jako například zmenHesloAdmina nebo login, vrací výsledek jen pokud je zadáno správné heslo. V opačném případě vyvolávají výjimku InvalidAdminPassword. Metoda vlozUzivatele vyhledá uživatele podle jeho přihlašovacího jména (login) a udělí mu přístup k zadanému účetnictví.

Další třída **Uzivatel** v sobě agreguje seznamy účetnictví a seznamy firem. Atributem je i instance UcetniPlatformy a to zejména z toho důvodu aby uživatel mohl přidělovat přístup k účetnictví jinému uživateli. Dalšími atributy jsou jméno, příjmení, login a heslo.

Třída **Ucetnictvi** může vzniknout nově (šablona implementována nebyla) a nebo z již existujícího účetnictví a automaticky posouvá účetní období o jedno vpřed. Agreguje

v sobě seznamy účtů a seznamy účetních operací. Implementovány byly metody na přidávání účtů a operací, které kontrolují správnost zadané operace a účtu. Metody pro odebírání a editování již implementovány nebyly.

Třída Firma obsahuje pouze relevantní informace o firmě, ke které se vede účetnictví

Třída **Obdobi** ukládá jak datum počátku účetního období, tak i dobu trvání tohoto období a to ve zlomcích roku. Metoda dalsi vrací novou instanci třídy Obdobi posunutou o dobu trvání.

Třída **UcetniOprace** obsahuje několik atributů důležitých k roztřídění účetních operací. Jde to datum, druh dokladu, číslo dokladu a kód. Pak také agreguje seznamy peněžních operací probíhajících na straně dal a straně má dáti.

Třída **PenezniOperace** obsahuje atributy jako částka, text a účet.

Třída **Ucet** obsahuje název, číslo a částku na straně dal a má dáti a pak implementuje některé metody pro aritmetiku s účty. Jde především o sčítání dvou účtů a přidávání částek na strany dal nebo má dáti.

7.3.2 Web

Internetová aplikace je implementována na sofistikovaném webovém frameworku Seaside a využívá tak jeho kompozitní strukturu.

Navržená aplikace se skládá ze dvou hlavních komponent – z uživatelské sekce a administrátorské sekce. Tyto dvě komponenty jsou zobrazovány současně. Dále se uživatelská sekce dělí na sekci seznamu firem a na sekci účetnictví, která je dále nahrazována sekcí vedení účetnictví (viz obrázek).

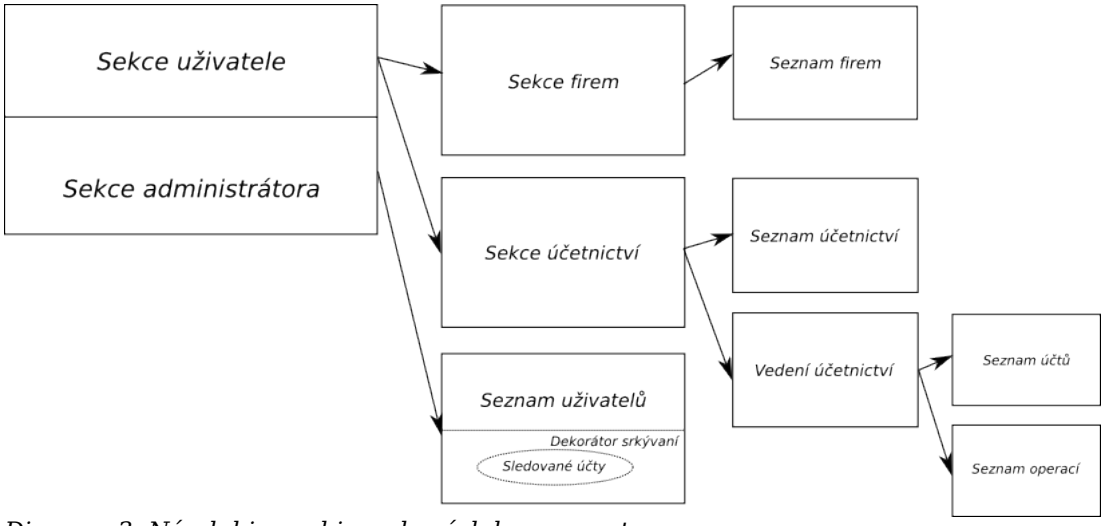


Diagram 3: Návrh hierarchie webových komponent

Model tříd odvozených z tříd Seaside pak vypadá takto:

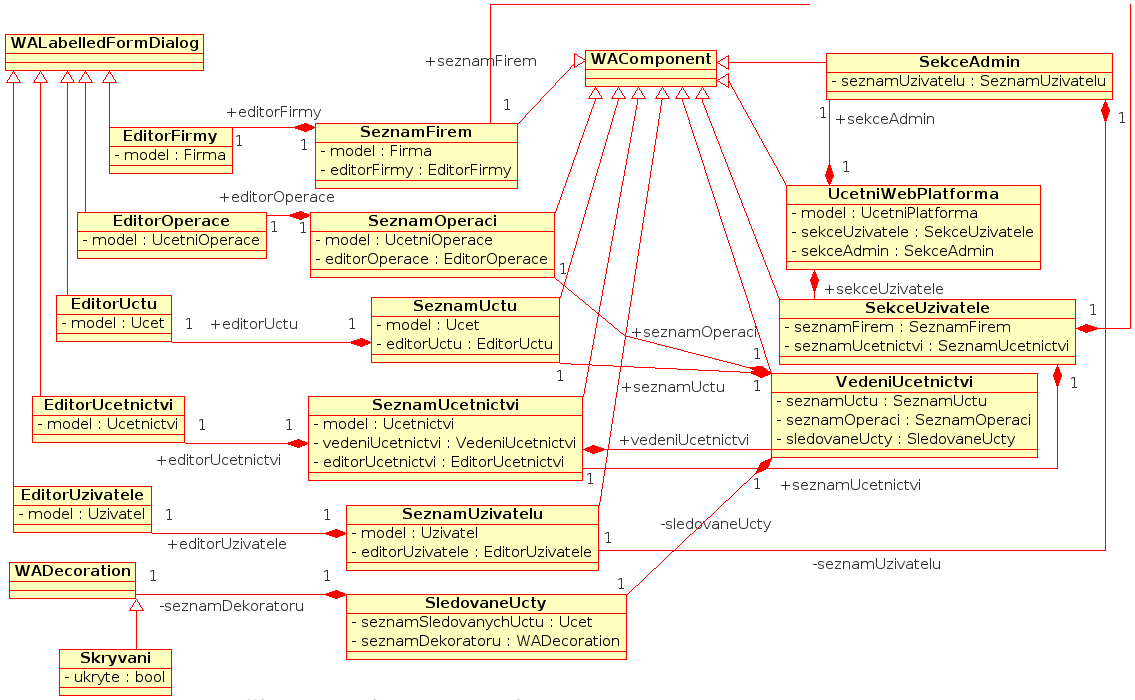


Diagram 4: Model tříd webového rozhraní

Většina tříd dědí z třídy WAComponent, kde přetěžují především metodu renderContentOn:, jež se volá při vykreslování konkrétní komponenty na webovou

stránku. Základní, neboli kořenovou komponentou je třída UcetníWebPlatforma, která je také registrována do hierarchie Seaside příkazem:

```
UcetniWebPlatforma registerAsApplication: 'Uctovani'
```

Ostatní třídy jsou agregovány, jak je znázorněno na schématu.

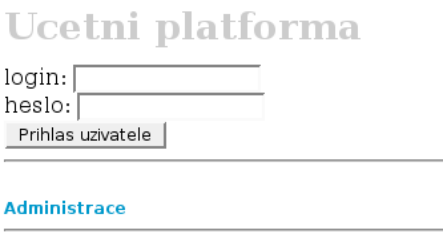
Implementován byl také jeden dekorátor Skryvani umožňující vlastníkovi ukrytí.

Pro uzpůsobení vzhledu byly aplikovány kaskádové styly v přetížené instanční metodě style kořenové komponenty – tedy UcetniWebPlatforma.

Popis webové aplikace

Aplikace je zaregistrována pod názvem Uctovani a její adresa může být například tato: http://localhost:9090/seaside/Uctovani. Po zadání tohoto odkazu se vytvoří instance třídy UcetniWebPlatforma a je také vytvořen model celé účetní platformy. Ten se následně uloží do třídní proměnné, takže dalším přístupem se bude pracovat nad stejným modelem.

Uživatel je na hlavní stránce vyzván k zadání jména a heslo. Vidí také odkaz s možností administrace:

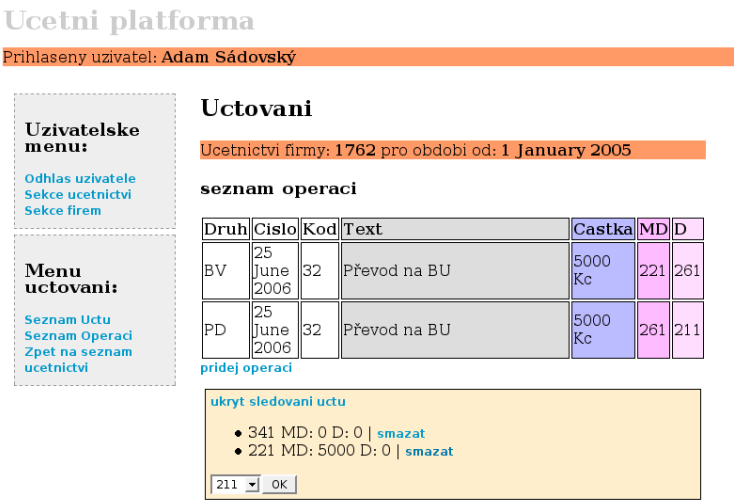


Po zadání jména a hesla a stisknutí tlačítka „Prihlas uzivatele“ vstupuje do systému. Zadal-li špatně některý z údajů, je informován a vyzván o opětovnému zadání:

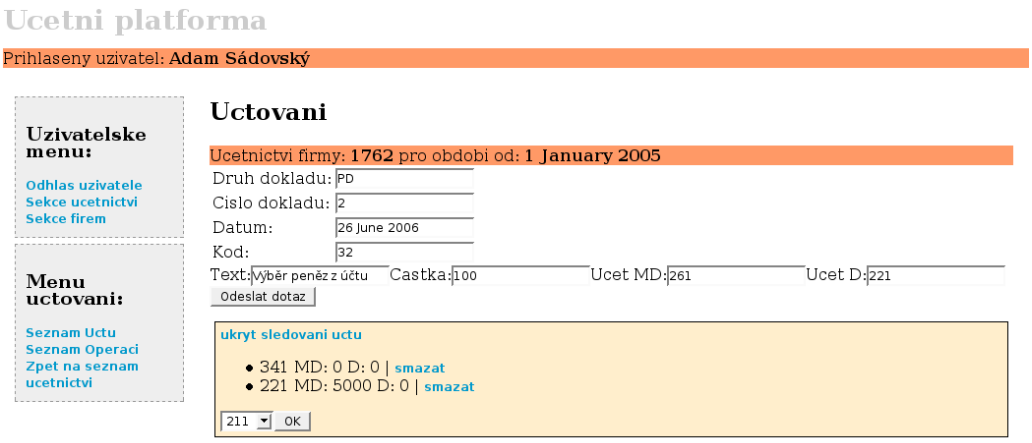
Po přihlášení je mu nabídnuto menu a seznam firem. Přitom má stále možnost vstoupit do administrace:

Po vstupu do sekce účetnictví:

Při vybrání účetnictví ve kterém chce uživatel účtovat, se objeví nové menu, ze kterého jsou přístupné seznamy účtů a seznamy účetních operací. Dále také může zobrazit stavy na účtech, které se mění, podle nově přidávaných operací. Kolem tohoto seznamu je obalený dekorátor, umožňující jeho skrývání:



Když uživatel zvolí přidání, či editaci nějaké položky, objeví se na místě seznamu položek podobný formulář:



8 Závěr

Nelze tvrdit, že webové aplikace a brány, jsou nejlepšími přístupy k jednotnému internetovému rozhraní pro sdílení, dat, informací a zdrojů. Existují i technologie založená na distribuovaných objektech, které nabízejí mnohem větší kontrolu nad tokem informací. Jde na příklad o projekt Crocquet (http://www.opencroquet.org/index.html). Ale webové aplikace mají i mnoho kladů, díky kterém jsou dnes velice populární a to především v souvislosti s rozvojem internetu.

Možností jak postavit webovou aplikace je mnoho a záleží především na využití. K dipozici jsou jak komerční, tak i open source nástroje pro vývoj sofistikovaných internetových aplikací.

Zdá se, že blízká budoucnost patří hlavně velkým frameworkům a to především těm, využívajícím objektové přístupy. Ty totiž dovolují mnohem abstraktnější pohled a kontrolovatelnější vývoj. V kurzu jsou i slova jako modulárnost, scalabilita či znovupoužitelnost.

Detailněji zmíněn byl například projekt Seaside, který možná ukáže cestu budoucímu vývoji.

9 Seznam literatury

1. Merunka Vojtěch, Pergl Robert, Pícka Marek, *Objektově orientovaná tvorba softwaru*, ČZU Praha, 2004, ISBN 80-213-1159-2
2. *Wikipedie: Otevřená encyklopedie: Object-oriented programming language* [online]. c2005 [citováno 19. 03. 2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Object-oriented_programming_language>
3. Jiří Kosek: *PHP – tvorba interaktivních internetových aplikací, podrobný průvodce*, GRADA Publishing, spol. s r.o., 1999, ISBN 80-7169-373-1
4. *Wikipedie: Otevřená encyklopedie: PHP* [online]. c2005 [citováno 19. 03. 2006]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Php>
5. *Wikipedie: Otevřená encyklopedie: PHP* [online]. c2005 [citováno 19. 03. 2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/PHP>
6. Mark Grand: *Java – Referenční příručka jazyka*, Computer Press, 1998, ISBN 80-7226-071-5
7. *Wikipedie: Otevřená encyklopedie: Java programming language* [online]. c2005 [citováno 16. 03. 2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Java_programming_language>
8. *Wikipedie: Otevřená encyklopedie: Smalltalk* [online]. c2005 [citováno 16. 03. 2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Smalltalk>
9. *Smalltalk.org: The early history of Smalltalk* [online]. [citováno 16. 03. 2006]. Dostupný z WWW: <http://www.smalltalk.org/smalltalk/TheEarlyHistoryOfSmalltalk_Abstract.html>
10. *Wikipedie: Otevřená encyklopedie: VisualWorks* [online]. c2005 [citováno 16. 03. 2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/VisualWorks>
11. *Wikipedie: Otevřená encyklopedie: Squeak* [online]. c2005 [citováno 16. 03. 2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Squeak>
12. *Squeak.org* [online]. [citováno 16. 03. 2006]. Dostupný z WWW: <http://www.squeak.org/>
13. *Wikipedie: Otevřená encyklopedie: Python* [online]. c2005 [citováno 16. 03. 2006]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Python>
14. *Wikipedie: Otevřená encyklopedie: Comparison of web servers* [online]. c2005 [citováno 17. 03. 2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Comparison_of_web_servers>

15. *Apache software foundation* [online]. c2005 [citováno 17. 03. 2006]. Dostupný z WWW: <http://www.apache.org/>
16. Scot Hillier, Daniel Mezick: *Programování Active Server Pages,* Comuter Press, 1998, ISBN 80-7226-118-5
17. *Wikipedie: Otevřená encyklopedie: Web framework* [online]. c2005 [citováno 26. 06. 2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Web_framework>
18. *The Web Framework Map* [online]. c2005 [citováno 25. 06. 2006]. Dostupný z WWW: <http://www.reahl.org/wfmwiki>
19. *Wikipedie: Otevřená encyklopedie: List of web application frameworks* [online]. c2005 [citováno 26. 06. 2006]. Dostupný z WWW: <http://en.wikipedia.org/wiki/List_of_web_application_frameworks>
20. *Root: Seaside – první díl* [online]. c2005 [citováno 25. 06. 2006]. Dostupný z WWW: <http://www.root.cz/clanky/seaside-1/>
21. *Seaside* [online]. c2005 [citováno 25. 06. 2006]. Dostupný z WWW: <http://seaside.st/>
22. *A Seaside tutorial by David Shaffer: Introduction* [online]. c2005 [citováno 25. 06. 2006]. Dostupný z WWW: <http://www.shaffer-consulting.com/david/Seaside/Introduction/index.html>