VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTERAKTIVNÍ TVORBA FORMULÁŘŮ V JSP

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

VLASTIMIL KALUŽA

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTERAKTIVNÍ TVORBA FORMULÁŘŮ V JSP

NÁZEV DIPLOMOVÉ PRÁCE ANGLICKY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE VLASTIMIL KALUŽA
AUTHOR

VEDOUCÍ PRÁCE Ing. RADEK BURGET, Ph.D.
SUPERVISOR

BRNO 2007

Abstrakt

Tato práce se zabývá vytvořením aplikace pro vytváření dynamických webových stránek propojených s databází. Zvláště na automatizovaný způsob vytvoření základního formuláře pro danou tabulku v databázi. Dále se soustředí se na vytyčení základních součástí aplikace s vizuálním návrhem a vytvoření diagramu tříd daného programu. Cílová aplikace má sloužit k usnadnění vytváření dynamických internetových stránek pro uživatele z jiných oblastí než informační technologie. Výsledná aplikace je určena pro další možné rozšiřování a vylepšování.

Klíčová slova

databázová aplikace, wysiwyg editor, vizuální návrh, editor webových stránek, generátor kódu

Abstract

This project is interested in creating application for generating dynamic web pages which are connected to database. Main intention of this project is concentrated to set out basic parts and create class diagram of this application. Final application will be used for simplification of web pages design for non-IT users. Final application represents concept for any extension and modification in the future.

Keywords

database application, wysiwyg editor, visual design, web page editor, code generator

Citace

Kaluža Vlastimil: Interaktivní tvorba formulářů v JSP. Brno, 2007, diplomová práce, FIT VUT v Brně.

Interaktivní tvorba formulářů v JSP

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vlastimil Kaluža
22. května 2007

Poděkování

Děkuji svému vedoucímu diplomové práce Ing. Radkovi Burgetovi, Ph.D. za trpělivost a odborné vedení diplomové práce.

Obsah

1 Úvod

Internet je slovo používané v dnešní době ve všech možných situacích. Každý, kdo chce najít nějaké informace, musí navštívit stránky na internetu nebo je na ně odkazován. V poslední době většina reklam informuje pozorovatele na internetové stránky produktů nebo firem, kde najdou mnohem více informací, než se vejde do reklamy. Na druhou stranu někteří uživatelé se stávají náročnějšími na vzhled a funkčnost internetových stránek. Je již doba dávno zapomenutá, kdy stačily čistě textové stránky. V dnešní době převládají dynamicky generované systémy, z nichž mnohé pracují i s databázemi.

Tvorba takových systémů však vyžaduje specifické znalosti a i pro získání základních informací musí autor nastudovat mnoho literatury. Existuje i mnoho různých technologií, které lze využít, a každá z nich má nějaké své výhody a nevýhody. Pro zvolení té správné technologie se musí člověk probrat dalším kvantem informací.

Záměrem tohoto projektu je usnadnění práce při vytváření dynamicky generovaných stránek pro běžného uživatele, kterých je stále více. Ideálním řešením je vytvoření aplikace, která bude schopna vytvářet dynamické systémy internetových stránek bez napsání jakéhokoliv kusu zdrojového kódu. Sice je to řešení utopické, ale dá se k němu přiblížit.

Výsledkem je dosažení aplikace podobné integrovaným vývojovým prostředím používaná pro implementační jazyky desktopových aplikací. Tyto systémy jsou velmi uživatelsky přívětivé a i běžný uživatel se je snadno naučí ovládat. Součástí těchto systémů bývají wysiwyg editory, které usnadňují vytváření a konfigurování vlastností jednotlivých komponent. Tyto editory ukazují výsledek již při postupném vytváření. Zkráceně řečeno: „Co vidím, to dostanu“.

V počáteční části práce chci poukázat na rozmanitost vývojových technologií a programovacích jazyků, které jsou v dnešní době hojně užívány pro vytváření dynamického obsahu internetových stránek. Tato část bude pokračovat navazující kapitolou o technologiích a teoretických znalostech, které jsou potřebné a použité pro řešení této diplomové práce.

Větší část práce zabere popis řešení ukázkové aplikace a rozbor diagramu tříd, kde se zmíním o záludnostech a rychlých, jednoduchých řešení těžkých úkolů, které stály přede mnou. Ukázkovou aplikaci této práce jsem si dovolil zkráceně nazvat WebEditor.

K závěru práce bych chtěl vyznačit další možné postupy vylepšování, popřípadě nové možnosti, které lze využít a objevili se v průběhu řešení zadání.

2 Technologie dnešní doby

Uživatel, který se v dnešní době chystá tvořit interaktivní internetové stránky je postaven před veliký úkol. Musí se rozhodnout, v čem chce tvořit stránky. Většina lidí se naučí jednu technologii, popřípadě jeden programovací jazyk, který dále prohlubují. Větší problém má člověk, který se učí tvořit stránku tzv. od nuly. V této situaci se většinou obrací na různá fóra, emailové konference, pomoc kamarádů a spolupracovníků. Informací je mnoho a mnohdy si odporují.

2.1 Programovací jazyky na straně serveru

Programovací jazyky, ve kterých lze v dnešní době tvořit dynamické stránky, můžeme rozdělit na několik skupin, podle několika pravidel. Prvním kritériem je způsob zpracování zdrojového kódu. Podle tohoto kritéria můžeme rozdělit programovací jazyky na následující skupiny:

- **Interpretované** – zdrojový kód se zpracovává při každém novém požadavku a po zpracování se teprve vykoná. Tím dochází k určitému zpoždění odpovědi a zpomalení vykonávání kódu. Výhodou je nezávislost na operačním systému a hardwarové platformě. Do této skupiny můžeme zařadit například PHP, Perl, Python, TCL a další.
- **Kompilované** – zdrojový kód se nejdříve přeloží do binární podoby. Toto je fáze zpracování a výsledný soubor se teprve distribuuje a je přímo vykonáván. Nedochází tedy k žádnému zpoždění. Nevýhodou je závislost na hardware a operační systém. Zde řadíme do skupiny například C/C++, Delphi.
- **Kompilované/Interpretované** – předchozí dvě skupiny měly své pro i proti. Proto někteří tvůrci vývojových platforem sáhly po smíšeném řešení. Zpracování zdrojového kódu je zdlouhavé, proti době jeho vykonávání. Proto se v této skupině zdrojové kódy přeloží, ale ne do binárního kódu, který je přirozený pro procesor, ale do mezikódu. Mezikód, u Javy bytecode a u .NET platformy MSIL (Microsoft Intermediate Language), je podobný tří adresnému kódu, který se dá již velice rychle interpretovat. Všechny již zmíněné platformy podporují různá vylepšení jako zpracovávání postupně za běhu programu, odkládání výsledků do vyrovnávacích pamětí a jejich znovupoužití. Pro nasazení na různé hardwarové platformy a operační systémy stačí pouze naprogramovat dané běhové prostředí.

Většina programovacích jazyků je již objektově orientovaná. U interpretovaných se objektově orientované programování objevilo v posledních pěti letech. U kompilovaných se tento fenomén objevil mnohem dříve a daly za vznik plně objektově orientovaným běhovým prostředím jaká jsou právě Java a .NET.

2.2 Značkovací jazyky a vzhled

Nesmíme zapomínat, že pouhá znalost programovacího jazyka nám nestačí. Internetové stránky se vytvářejí podle HTML/XHTML standardu. HTML/XHTML je značkovací jazyk, který vytváří strukturu (kostru) dokumentu, který nám prohlížeč zobrazí. Programovací jazyk nám pouze umožní reagovat na události vzniklé na dané stránce a vygenerovat regulérní obsah pro uživatele. Tedy další jazyk, který uživatel musí umět pro tvorbu stránek.

Líbivost stránek se docílí pomoci designu, který můžeme zapisovat již do XHTML kódu, nebo do externího souboru, který se nazývá kaskádový styl (CSS). Kaskádové styly se liší, tedy jejich implementace pro různé internetové prohlížeče. Je to důsledek původního nestandardizovaného vývoje protokolu HTML.

2.3 Klientské programování

A jako poslední v řadě zmíním klientský skriptovací jazyk. Tedy programovací jazyk, který umožňuje pracovat se stránkou na straně klienta a tím zvýšit kvalitu webové prezentace. Ano mluvím o JavaScriptu. Tento skriptovací jazyk v posledních letech značně nabral na důležitosti při vývoji internetových stránek a to díky knihovně AJAX. Tato knihovna obsahuje funkce a třídy pro komunikaci mezi klientem a serverem na pozadí a to aniž by si uživatel, který si dané stránky prohlíží, všimnul, že dochází k získávání nových dat nebo částí stránek. AJAX umožňuje přiblížit webové aplikace k desktopovým aplikacím, jednak svým vzhledem, ale hlavně svou funkčností.

Tímto krátkým výčtem jsem chtěl jen upozornit na obecné klady a zápory některých technologiích a jak je vidno, je jich požehnaně mnoho a každá má své pro a proti. Pro tvorbu musíme umět nejen nějaký ten programovací jazyk, ale taky znát kód HTML/XHTML, CSS, JavaScript. Zkráceně řečeno čtyři programovací jazyky pro začátek.

Cílem tohoto zadání je právě přiblížit tvorbu webových aplikací obyčejným uživatelům bez potřebných počátečních znalostí. K tomu je zapotřebí naučit tyto znalosti aplikaci, která má takovéto stránky tvořit.

3 Použité technologie

Pro vytvoření aplikace jsou vybrané volně dostupné a rozšířené implementační technologie. Volně dostupné technologie zaručují dobrou podporu a mnoho informací na internetu s řešením různých problémů. Volně dostupné technologie bývají nejrozšířenější, protože jejich rozšiřování nebrání žádná přísná licenční politika, jak je tomu u komerčních produktů.

Aplikace bude implementována v plně objektově orientovaném programovacím jazyce Java, který se právě kompiluje pouze do bytecodu a ten je teprve vykonáván. Technologie JSP je zvolena pro vytváření obslužných kódů webových stránek v aplikaci. Po nasazení na server se tento kód vykonává na straně serveru a výstup tohoto kódu se odesílá klientovi. Pro definování vzhledu internetové stránky je vybrán značkovací jazyk XHTML, který je standardizován a je vyspělejší než jeho předchůdce HTML. Aplikace spolupracuje s databází a jako výchozí databázový systém byla vybrána databáze MySQL.

3.1 Java

Programovací jazyk Java byl vyvinut společností SUN Microsystems. Tento jazyk, spíše bychom měli říkat platforma, zasahuje od mobilních zařízení přes klasické počítače až k velkým serverovým řešením. Dle toho se také dělí na jednotlivé distribuce. Na stránkách http://java.sun.com můžeme stáhnout i další podpůrný software, k dalšímu rozšiřování této platformy. Existuje mnoho třetích stran, které vytvářejí aplikace a dokonce celá vývojová prostředí a sady. Tím vším se z Javy jako programovacího jazyka stala v posledních letech technologie, která je opravdu velmi obsáhlá a sahající do všech zákoutí informačních technologií.

3.1.1 Vlastnosti Javy

Java, i celá technologie, která je nad ní vystavěna, je nezávislá na hardwarových platformách i na operačních systémech. Všechny aplikace vytvořené v této technologii neběží přímo na procesoru, protože výsledkem kompilace zdrojových textů není známý nativní kód[1], jako je to u C/C++, ale bytecode[2], který má různá další rozšíření. Můžeme si ho představit jako assembler s mnohými vylepšeními. Přesněji se jedná o tří adresný s kód s podporou výjimek, tříd, zabezpečení, správu paměti a událostí.

Tento kód, můžeme užít i mezikód, jak je tomu u platformy .NET (Intermediate Language), potřebuje svůj vlastní virtuální procesor nazývaný běhovým prostředím. Je to aplikace, která je již

[1] Nativní kód je binární kód, kterému rozumí daný procesor, na kterém tento binární kód vzniknul.

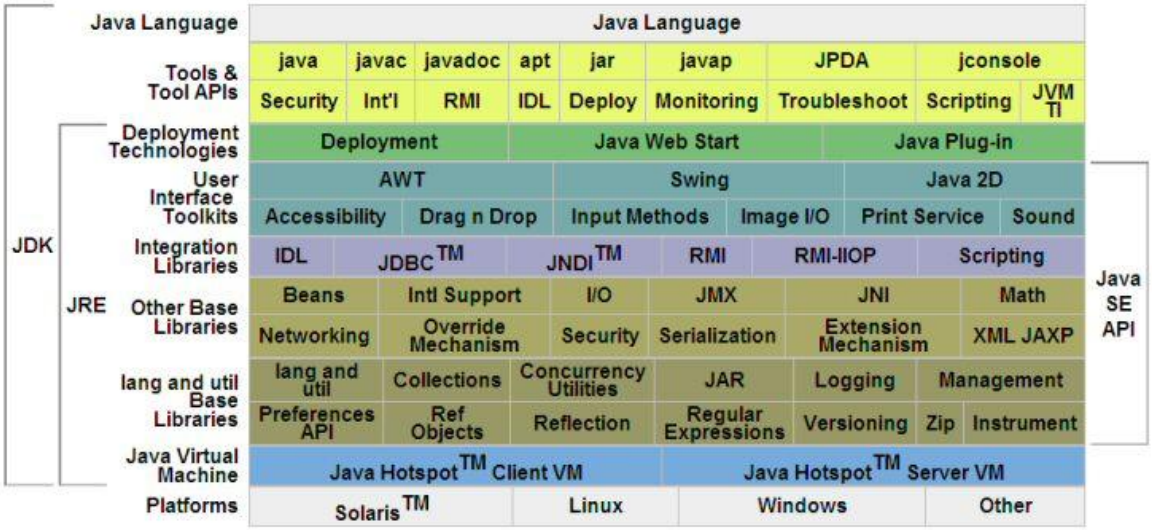
[2] Bytecode (bajtkód) je složenina dvou anglických výrazů byte a code a znamená to kód, který už není čitelný pro člověka, ale také ne pro procesor.

přeložena do nativního kódu dané hardwarové platformy a operačního systému. Díky tomuto běhovému prostředí může být bytecode vykonán. Toto běhové prostředí (runtime framework) vytváří stále stejné podmínky na různých platformách hardwaru a softwaru. Tím jsou aplikace napsané v Javě přenositelné a rozšířitelné na všechny počítače a nejen počítače, třeba i mobilní telefony. Tato rozšiřitelnost Javy byla podnětem k jejímu výběru.

Java je plně objektově orientovaným jazykem. To znamená, že na vrcholku všeho je jedna třída `java.lang.Object`. Třída se jmenuje `Object` a je uložena v balíku `java.lang.` Balíky nám slouží k lepší orientaci ve spoustě různých tříd, které Java nabízí. Důkaz plné objemovosti nám může být nejjednodušší program HelloWorld! napsaný v Javě:

```
public class Main{
    public static void main(String args[]){
        System.out.println("Hello World!");
    }
}
```

Základem je třída `Main`, která implementuje jednu metodu `main`, která je vstupním bodem celé aplikace.



Obrázek 1: Platformy Javy

3.1.2 Platformy Javy

Java, jak jsem se již zmínil, zasahuje do mnohých zákoutí informačních technologií a podle toho se taky musí vybrat správná platforma. Základní distribuce Javy jsou čtyři:

1) J2ME – SDK pro mobilní zařízení
2) J2SE – SDK pro počítače
3) J2EE – SDK pro serverové systémy
4) J2RE – běhové prostředí pro všechny distribuce

Uživatel, který chce pouze používat aplikace napsané v Javě, nemusí mít právě nainstalovanou nějakou celou platformu (SDK – Software Development Toolkit, Nástroje pro vývoj softwaru), ale vystačí jenom s virtuálním procesorem neboli běhovým prostředím (runtime framework). Toto prostředí se dá stáhnout ze stránek Javy a slouží pouze ke spouštění aplikací. V dnešní době je JRE již ve verzi 1.6.0_01. Ostatní SDK balíky obsahují nástroje a sadu knihoven pro vyvíjení aplikací na daných platformách. Dokonce lze stáhnout i některá celá vývojová prostředí, například Netbeans 6.0, WebSphere, Eclipse 3.2.1, popřípadě jeho balík Calisto a další.

3.2 JSP – Java Server Pages

JSP je jedna z mnohých částí technologie Javy. Slouží, stejně jako servlety (*viz. kapitola* **Chyba! Nenalezen zdroj odkazů.**), k dynamickému zpracovávání a k vytváření internetových stránek. JSP můžeme chápat jako využití programovacího jazyka Java na webových stránkách obdobně jako skriptovací jazyky PHP nebo ASP. Na rozdíl od těchto skriptovacích a pseudo-objektových jazyků je JSP plně odvozeno od Javy a tím nám přináší plně objektový přístup k programovacímu rozhraní, které je v dnešní době hlavním kritériem.

Další velkou výhodou je používání klasických tříd dostupných v Javě, jejichž autoři nemají někdy ani ponětí, že je takto někdo někdy bude využívat k vytváření stránek v JSP. Tím máme možnost využít celou sadu knihoven, které nám Java nabízí. Zajisté je nám jasné, že i autoři a tvůrci Javy mají o mnoho méně práce při vytváření knihoven.

Pokud již někdo něco programoval na internet a také nějakou desktopovou aplikaci, určitě ví, že se musel učit podruhé nový způsob programování nebo styl programování. Jednoduše nemohl využít znalostí, které již měl z předchozích naprogramovaných aplikací. U Java Server Pages a Javy je tomu jinak. Tím, že máme stejný programovací jazyk a používáme ho jak pro webové aplikace, tak i pro desktopové aplikace, můžeme získané zkušenosti znovu a znovu uplatňovat, aniž bychom se museli znovu učit novou věc.

3.2.1 Rozdíly od ostatních jazyků

V předcházejících odstavcích jsem zmínil slovíčko „servlet“, které patří již delší dobu k působení Javy na internetu. Teď bych se chtěl k němu vrátit a trochu jej objasnit. Servlet je programová podoba HTML dokumentu. Je to jedna z dalších částí Javy jak jsem již dříve zmiňoval. Nejlepší vysvětlení bude na příkladě. Prvním příkladem bude stránka JSP, která vypíše aktuální datum.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs">
    <head>
      <title>Dnešní datum</title>
    </head>
    <body id=“datum“>
      Dnešní datum: 
      <br />
      <%= new java.util.Date(); %>
     </body>
   </html>
```

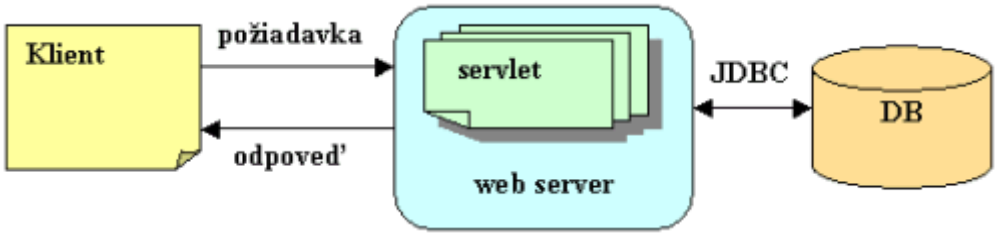
Z uvedeného příkladu je zřejmé, že JSP stránka je klasická stránka napsaná pomocí značkovacího jazyka XHTML. Programový kód napsaný v Javě, který reprezentuje výkonnou část JSP, je do tohoto textu vložen pomocí skriptovacích značek `<%= %>`. Tyto a další jiné značky si vysvětlíme později v textu. Nyní se podíváme, jak stejná stránka vypadá, když ji napíšeme pomocí servletu.

```
package datum;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;


public class DatumServlet extends HttpServlet {
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML><HEAD><TITLE>Dnešní datum</TITLE></HEAD><BODY>");
    out.println(new Date().toString());
    out.println("</BODY></HTML>");
    out.close();
  }

  public String getServletInfo() {
    return "DatumServlet 1.0";
  }
}
```

Je vidět, že servlet je přesným opakem JSP, neboli je to program, který vygeneruje HTML dokument a poté jej zašle klientovi. Výsledkem servletu je zkompilovaný kód, který se vykonává na serveru daleko větší rychlostí, než veškeré ostatní skriptovací jazyky. Samozřejmě se jedná o bytecode. Proč tedy jedna platforma přináší dva přístupy? Je to dáno tím, že servlety potřebují své vlastní podmínky a nastavení pro to, aby správně běžely. To je oproti JSP určitá nevýhoda. Stránky JSP se totiž při spuštění, neboli obdržení požadavku od klienta na serveru, přeloží na servlet, který je poté zpracován serverem a výsledek odeslán klientovi zpět. Stránky JSP nemusíme umísťovat do speciálních adresářů jako servlety, ale mohou být mezi ostatními HTML soubory. Nepotřebujeme ani jiná další specifická nastavení.

3.2.2 Průběh zpracování JSP

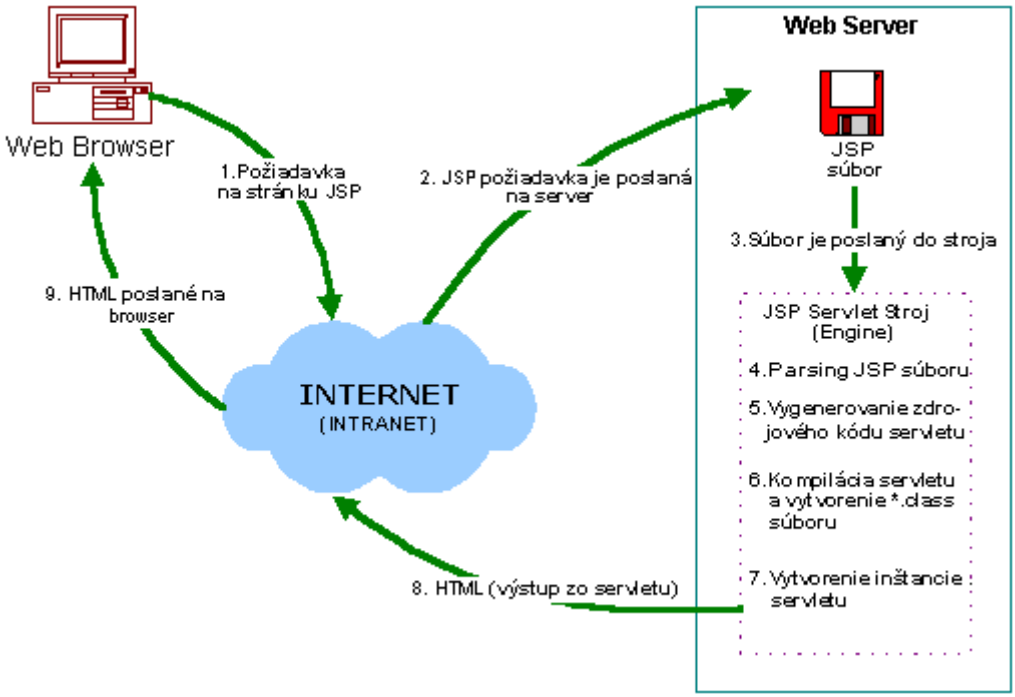
Jak tedy dochází ke zpracování JSP stránky? Vezmeme to popořadě. Ukážeme si, jak se zpracovává servlet, viz následující obrázek.



Obrázek 2: Průběh zpracování dotazu na servlet [1].

Servlety jsou umístěny na webovém serveru ve speciálních adresářích, kde můžou být spuštěny a server k nim má přístup. Klient se na ně může dotázat pomocí HTTP požadavku. Servlet se zpracuje a výsledek je odeslán zpět klientovi jako HTTP odpověď.

U stránek JSP je situace trochu komplikovanější. Stránka JSP se musí přeložit na servlet a ten se zpracuje již známým způsobem. K překladu slouží JSP Servlet engine, stroj k vytvoření servletu a jeho zkompilování na běžící aplikaci. Pak již obdržíme výsledek.



Obrázek 3: Průběh zpracování dotazu na JSP stránku [2].

3.2.3 Skriptovací značky a direktivy JSP

JSP má čtyři různé skriptovací značky. Dělí se podle toho, kde se kód, uzavřený těmito značkami, vloží do výsledného servletu. Jsou jimi:

<%= %> - výraz

<% %> - skriptlet

<%! %> - deklarace

<%-- --%> - komentář

Nyní si je trochu popíšeme.

3.2.3.1 Výraz

Výrazem se chápe jakýkoliv příkaz, který se má vložit do výstupu na stránku. V rámci servletu je to parametr funkce `out.print()` třídy `java.io.PrintWriter`. Nejčastějším použitím této skriptovací značky bude při vložení hodnoty nějaké proměnné do výstupu stránky. Příklad zápisu můžeme vidět v následující ukázce:

```
<%= new java.util.Date(); %>
<%= request.getRequestURI()%>
```

Středník na konci může a nemusí být.

3.2.3.2 Skriptlety

Skriptlety jsou kusy kódu, které se vkládají do metody servletu `_jspService()`. Jako výrazy mají přístup k implicitním objektům (`request`, `response`, `session`, `out`, `atd`.), ale mohou být od nich daleko složitější. Pokud potřebujeme něco vypsat ze skriptletu, musíme k tomu použít objekt `out`. Příkladem skriptletu může být následující úsek:

```
<%
String method = request.getMethod();
out.println("Method: " + method);
%>
```

3.2.3.3 Deklarace

Už jen z názvu pojmu deklarace je jasné k čemu slouží – pro vytváření nových metod a proměnných. Nic nevrací na výstup stránky, a proto se používají spolu s výrazy. Vkládají se mimo metody `_jspService()`. Příkladem může být:

```
<%!
private String helloWorld = "Hello World!";
public String getString() {
        return helloWorld;
}
%>
```

3.2.3.4 Komentáře

Posledním jsou komentáře JSP, které se ve výsledku neobjeví nikde. Jsou součástí pouze zdrojových kódů stránek JSP. Slouží pro lepší orientaci v kódu, zvláště, když se ke kódu vrátíme po delší době. Příkladem komentáře je:

```
<%-- komentář --%>
```

3.2.3.5 Direktivy

Kromě skriptovacích značek můžeme používat, a někdy i musíme, direktivy. Ty slouží například pro vkládání tříd z knihoven nebo celých souborů. Také nám budou nápomocny při vytváření vlastních značek. Jejich zápis vypadá následovně:

```
<%@ direktiva ... %>
```

Máme k dispozici celkem tři typy direktiv. Přehledně jsou uvedeny v následujícím výčtu:

`page` – slouží pro importování knihoven, nastavení vlastností stránek (obsahu, kódování, povolení nebo zakázání session)

`include` – slouží ke vkládání obsahu dalších celých souborů. Tím si můžeme stránku rozdělit na dynamický a statický obsah a lépe se potom v kódu orientovat.

`taglib` – slouží k definici nových uživatelských značek.

Pro řešení projektu budeme potřebovat pouze první dvě direktivy. Pomocí `page` budeme importovat knihovny z Javy. A `include` nám poslouží pro vkládání souboru například s připojením do databáze.

3.3 XHTML

Tento značkovací jazyk je kombinace dvou značkovacích jazyků. Pro znalejší lidi v oboru, již sám název připomíná, jaké jazyky to jsou. Správně je HTML a XML. Proč? Proč nový značkovací jazyk? HTML už nestačí? To jsou otázky, které napadnou asi každého programátora. Důvodů pro nový jazyk je několik. Tím hlavním důvodem je, že jazyk HTML se vyvíjel postupně s prohlížeči a každý tvůrce prohlížeče si do něho přidával své vlastní značky. Díky tomu můžeme vidět stránky různě v různých prohlížečích.

Standardizovat jazyk HTML je nadlidský úkol a tak organizace World Wide Web Consorcium (W3C) od toho záměru upustila. Nová doba používá čím dál tím více jazyk XML a to v různých směrech. Kombinací XML s HTML vznikl jazyk XHTML, který je již standardizován a pokud prohlížeče chtějí tyto stránky zpracovávat, musí se standardu přizpůsobit. Jazyk XHTML se dělí na tři varianty podle toho, jak moc je benevolentní vůči staršímu HTML. Dnes je k dispozici verze 1.1 a to ve variantách:

- Strict
- Transitional
- Frameset

V dnešní době už můžete na stránkách W3C (http://www.w3c.org) najít draft verze 2.0.

Proč tolik variant XHTML? Je to z důvodu způsobu vytváření stránek a jejich zpětné kompatibility. Varianta strict, která je nejpřísnější, se drží čistě specifikace XHTML a nedovoluje žádná další rozšíření z obyčejného HTML. Pokud se ovšem nechceme některých starších způsobů vzdát, můžeme použít verzi transitional. Verze frameset odpovídá verzi předchozí transitional, jenom navíc povoluje použití rámů.

3.3.1 Rozdíly mezi XHTML a HTML

Hlavním rozdílem mezi XHTML a HTML je začlenění značkovacího jazyka XML. To znamená, že každá značka má svoji ekvivalentní ukončovací protiklad tak jak jsme byli zvyklí u některých značek v HTML. U XHTML je to již u všech. I obyčejná značka `<br>` má svůj konec ve značce `</br>`. Samozřejmě, že takovéto značky lze zkrátit na varianty `<br/>` a `<br />`. Doporučuji poslední uvedený způsob, kvůli kompatibilitě starších prohlížečů, které značku `<br/>` přeskočí.

Všechny značky a názvy atributů musí být psány malými písmeny a všechny hodnoty atributů uzavřeny do uvozovek. Nepovoluje se atribut `name` u značek, ale u všech je vyžadován atribut `id`. Samozřejmě, každá varianta XHTML povoluje některé výjimky.

3.4 MySQL

MySQL je volně dostupná a výkonná databáze. Vystačí pro mnohé internetové aplikace. Je nejčastěji dostupná na webových serverech, kde si můžete pronajímat místo pro vaše dynamické webové aplikace (webhosting). MySQL je databáze rychlá, lehce konfigurovatelná a podporuje víceuživatelský přístup a základní principy práce nad relačními systémy. Samozřejmě, pro profesionální provoz a využití je možnost zakoupení komerční licence, kde navíc získáte podporu a servis.

Pro naše účely vystačí volně dostupná (Community) verze na internetu. V nedávné době vyšla verze MySQL 5.0. Chtěl bych upozornit na rozdílnost verzí předchozích a této verze. Rozdíly nejsou až tak zřejmé a většina starších aplikací by měla fungovat stejně. Ale můžou se vyskytnout problémy. Většina jazyků má již k dispozici dvě verze ovladačů k přístupu na databázi. Proto bychom měli nejdříve zjistit verzi systému, kterou budeme využívat.

V dnešní době je již dostupná verze MySQL 6.0. Ze svých vlastních zkušeností můžu doporučit začínajícím uživatelům verzi 5.0.3 a níž, které jsou po nainstalování ihned plně funkčně k dispozici. Tyto nižší verze mají implementované datové typy, které jsou kompatibilní i s verzemi 4.*. U vyšších verzí doporučuji server nakonfigurovat, sice pro neznalého to může být problematické, ale vývojáři MySQL vytvořili průvodce nastavením serveru. Pozor, server lze špatným nastavením i zablokovat, potom už nepomůže nic než reinstalace.

Ke správě serveru a databází můžeme využít terminálovou aplikaci dodávanou se serverem, nebo využít jeden z následujících programů. MySQL Front je aplikace určena pro Windows a pro uživatele je velmi přívětivá. Pokud chceme přistupovat i vzdáleně a odkudkoliv, doporučuji PhpMyAdmin. To je aplikace vytvořená pomocí skriptovacího jazyka PHP a obsahuje základní nástroje pro práci s tabulkami a celými databázemi. Lze umožnit i víceuživatelský přístup podle databáze. K běhu je potřeba nainstalován PHP a jeden z webových server Apache nebo Internet Information Server. Od tvůrců je možné stáhnout další grafické aplikace MySQL WorkBench, MySQL Query Browser a MySQL Administrátor, která je obdobou MySQL Frontu. Více připomíná prostředí pro správu Microsoft SQL serveru.

Pro připojení k databázím z Javy slouží knihovny tříd zvané konektory. Pro každý databázový stroj existuje příslušný konektor. Proto si musíme stáhnout ten správný konektor. Nejlépe ze stránek MySQL a pro správnou verzi databáze. Databáze i konektor je dostupný na stránkách http://dev.mysql.org/downloads. Zde máme k dispozici konektory pro různá rozhraní. Vyjmenuji jen

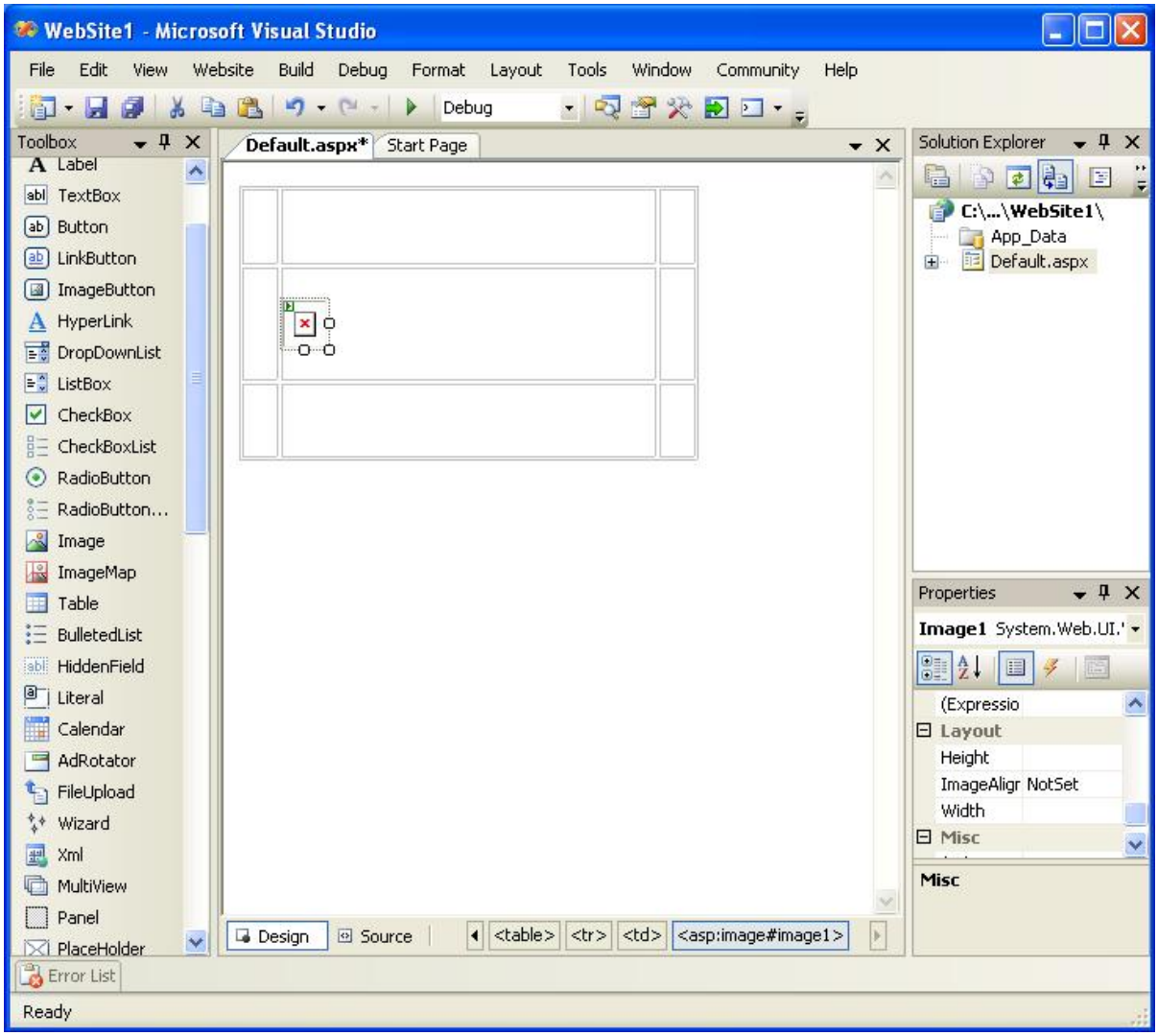
ty nejznámnější: ODBC, Java, .NET, PHP a další. Na stejné stránce jsou k dispozici i aplikace pro správu databáze například MySQL Administrátor.

4 Představa cíle

K vytvoření modelu aplikace jsem použil Rational Rose a modelovací jazyk UML. Výsledkem je rozšířený model tříd (class diagram), který si postupně probereme v kapitole „Návrh řešení“, protože model je rozsáhlejší.

Diagram se skládá ze 3 částí: Aplikační třídy
Projektové třídy
Komponentové třídy

Význam jednotlivých částí si vysvětlíme v následujících podkapitolách. Nesmíme zapomenout, že v modelu bude mnoho rozhraní, které se různě prolínají. Celý výsledný model tříd je součástí přílohy a je většího rozměru než A4. Cílem tohoto modelu by měla být aplikace podobná vývojovým prostředí pro .NET nebo NetBeans, viz obrázek č. 3.



Obrázek 4: Představa vzhledu a základních součástí aplikace. Uvedeno je prostředí Visual Studio.NET.

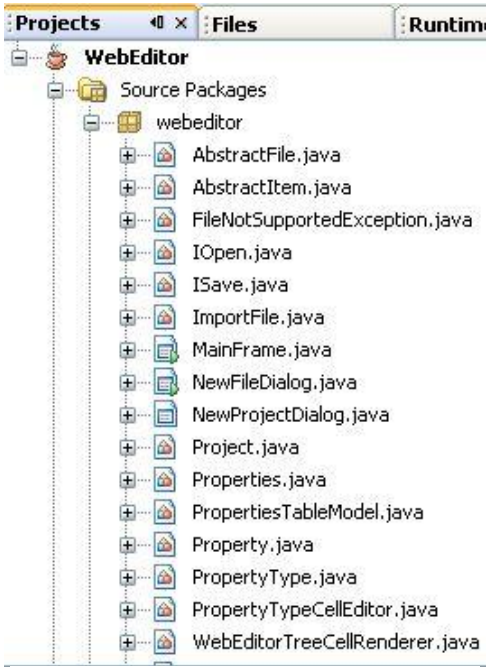
4.1 Aplikační třídy

Aplikačními třídami jsem nazval skupinu tříd, které budou vytvářet vzhled aplikace a obsluhu její funkčnosti. Budou to všechna dialogová okna vyvolána hlavním oknem aplikace. Hlavní aplikace bude obsahovat několik důležitých částí. Budou to:

- Strom projektu
- Tabulka vlastností
- Editor zdrojového textu
- Wysiwyg editor

4.1.1 Strom projektu

Bude obsahovat stav projektu. To znamená, že v jednom okamžiku bude rozpracovaný pouze jeden projekt. Tento projekt bude obsahovat soubory s internetovou stránkou (*.jsp soubor), nebo soubor určený ke vkládání (*.jsp soubor bez vzhledu). Každý soubor bude mít své jméno a cestu, které budeme moci měnit. Strom projektu se bude ukládat do zdrojového souboru projektu s příponou „.weprj“. Mezi importované soubory patří hlavně soubor s připojením do databáze.



Obrázek 5: Projektový strom projektu WebEditor v NetBeans 5.5

Strom projektu bude rozšířen o celkový strom vzhledu stránky. Tím se myslí to, co bude na stránce obsaženo, a který prvek bude obsahovat další prvky. Tento strom by měl být rozlišen různými ikonami dle typu uzlu (Projekt, Adresář, Soubor, Stránka, komponenta na stránce, …). S takovýmto rozšířením uživatel získá větší přehled o sestavené stránce a rychlý přístup ke všem elementům na stránce.

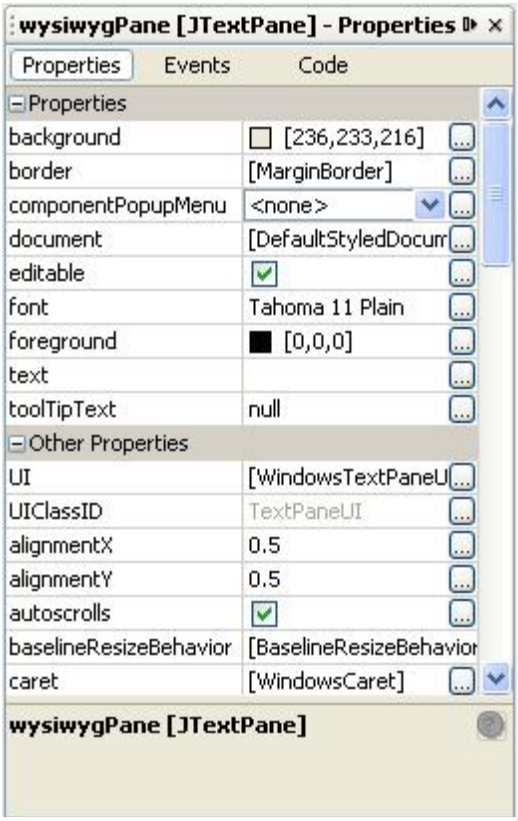
4.1.2 Tabulka vlastností

Tato tabulka bude umožňovat měnit vlastnosti jakýchkoliv objektů, které budeme mít vybrané. Tyto objekty, můžeme použít i pojmenování elementy nebo prvky, musí nejspíš v rámci modelu tříd implementovat rozhraní pro práci s vlastnostmi `IProperties`. Tabulka se bude skládat ze dvou sloupců, názvů a hodnot vlastností daného objektu.

Na první pohled se může zdát zbytečné vytvářet tuto tabulku. Nesmíme ale zapomenout, že aplikace bude obsahovat wysiwyg editor, který bude moci obsahovat různé objekty a jediným způsobem, kterým budeme moci měnit vlastnosti těchto objektů, je právě tabulka vlastností.

Pro rychlejší a přesnější práci se bude muset dodat několik nových funkcí. Budou to funkce pro rozlišení vlastností na kategorie, definovat typ vlastnosti, popřípadě umožnit zobrazit dialog pro podrobnější nastavení. Kategorie nám bude moci umožnit třídit vlastnosti dle jejich významu pro rychlejší hledání a nastavování. V případě, že uživateli se rozdělení nebude líbit, bude si moci vlastnosti seřadit dle abecedy.

Dále bude muset každá vlastnost definovat své výchozí hodnoty a celá aplikace bude muset umožnit jejich globální nastavení a pamatovat si je do další změny. Pro náročnější vlastnosti, například styl fontu, lze umožnit užití dialogových oken pro jejich sestavení.



Obrázek 6: Tabulka vlastnosti komponent JTextPane z NetBeans 5.5

4.1.3 Editor zdrojového textu

Bude to klasický textový editor jako je například NotePad ve Windows. Bude spíše sloužit k nahlédnutí do vytvářeného kódu. Tento kód je výsledkem druhého editoru, který by měl být primárně používaný uživatelem. Jak již bylo zmíněno v úvodu, snahou je vytvořit takovou aplikaci, aniž by uživatel musel zasahovat do zdrojového textu a tím se tato aplikace stala velmi uživatelsky přátelskou nejen pro začátečníky.

Ale i přes všechnu snahu, nebude moci aplikace dotvořit kód, popřípadě vzhled stránky do úplné dokonalosti. V takovém případě bude muset uživatel pozměnit vygenerovaný kód, popřípadě doplnit další bloky kódu. V takovémto případě bude nutné zajistit paměť na řádky kódu, které dodal uživatel, aby se při novém generování neztratili. Popřípadě najít informační zdroj, který o tomto blíže pojednává.

Jedná se hlavně o kód vzhledu stránky (XHTML), protože výsledný obslužný skript se po novém vygenerování může stát syntakticky, nebo sémanticky špatným.

4.1.4 Wysiwyg editor

Čím dál více aplikací začíná nabízet takzvané wysiwyg editory. Pod tímto pojmem si můžeme představit editor, který se ovládá pomocí myši (tzv. drag-and-drop) a jednotlivé objekty se spravují

výhradně touto periferií počítače. Slouží k vytváření grafického vzhledu, v našem případě stránek, ale mohou to být i vzhledy desktopových aplikací, appletů nebo grafických děl a modelů.

Zkráceně se dá pojem wysiwyg přeložit na: „Co vidím, to dostanu“. To znamená, že výsledek, který bude běžet na nějakém serveru, nám vrátí vzhledově stejnou stránku, kterou vidíme, když ji vytváříme. Právě proto bude zapotřebí tabulka vlastností, pomocí níž budeme moct nastavením vlastností dotvořit jeho vzhled. Jiný způsob editace jednotlivých objektů nebude již k dispozici. Ovšem pokud nebudeme brát změny pozice pomocí klávesových šipek a všeobecně vstupy z klávesnice.

Z reprezentace stavu editoru se vytváří zdrojový kód, který se bude kopírovat do textového editoru. Vlastnosti, které budeme moci nastavovat, se bude přenášet do tohoto textu a dotvářejí hodnoty jednotlivých parametrů. Tento kód se bude každou změnou v editoru měnit.

Samotný editor nám bude nabízet sadu komponent, které budeme moci vkládat na stránku. Každá komponenta bude mít své specifické vlastnosti určené pro úpravy a také svou reprezentaci do zdrojového textu. Tuto sadu, kterou bude nabízet aplikace přímo, můžeme nazvat základní sadou. Dalším vylepšováním se sada bude moci rozšiřovat a při správném návrhu budou moci i další strany vytvářet své vlastní komponenty.

Jelikož budeme vytvářet v převážné většině formuláře, budeme pro umisťování komponent užívat tabulky, kde do každé buňky můžeme umístit nejméně jednu komponentu. Ovšem, že to bude moci být opět tabulka a tak nám bude umožněno vytvářet rozsáhlejší a strukturně složitější tvary tabulky. V nejlepší variantě aplikace, by měl mít uživatel možnost zvolit si způsob základního pozicování komponent na dané stránce daného souboru. I když takováto volba je v tomto pohledu již prvním adeptem na rozšíření. V případě, že si zvolí tabulku, bude se vše řídit rozměry tabulky, pokud ovšem zvolí absolutní pozicování, nebude muset tabulku použít.

Pro ukládání projektu a jednotlivých jeho souborů se bude stav editoru uchovávat v externích souborech. Tato varianta je daleko jednodušší pro počáteční implementaci, než vytváření vzhledu pomocí procházení a rozkladu zdrojového textu. Takováto metoda by prodlužovala načítání jednotlivých souborů do aplikace a byla možná i samostatným ročníkovým projektem nebo diplomovou prací.

Vzhled se bude ukládat do externích souborů, které se budou jmenovat stejně jako soubor, který tento vzhled bude mít, s příponou `.jspr`. To znamená, pokud budeme mít soubor `index.jsp`, tak soubor vzhledu bude pojmenován `index.jspr`. Koncovka souboru je zvolena dle názvu souboru s jeho příponou a prvním písmenem z anglického slova „resource“ (zdroj).

Všechny čtyři zmiňované části budou umístěny v hlavním okně, které bych si představoval podobně jako vývojová prostředí Netbeans nebo Microsoft Visual Studio .NET. V hlavní nabídce by

měly být standardní možnosti pro práci s projektem a se soubory. Dále možnost vybírat jednotlivé komponenty na stránku wysiwyg editoru. Rozmístění by mělo být funkčně výhodné pro uživatele.

Z hlavního okna jsou dostupné dialogy pro otevření a vytvoření projektu a vkládání souborů do projektu. Také dialog nastavení celé aplikace musí být nedílnou součástí. Samozřejmostí jsou varovná hlášení při různých neúspěších a potvrzovací dotazy pro možnost ukliknutí se mimo.

4.2 Projektové třídy

Tyto třídy budou sloužit k obsluze a uchování souborů, ze kterých se bude skládat projekt. Soubory budou děleny dle typu. Například webové stránky, soubory pro vkládání, soubor s kaskádovým stylem a další. Podle toho bude muset třída reprezentující projekt obsahovat seznamy těchto souborů jakožto objektů.

Soubory mají svůj obsah, který se bude načítat ze souborů uložených na disku. Obsah některých se generuje z wysiwyg editoru, nebo se bude kombinovat z více zdrojů. Každá třída, která reprezentuje soubor, bude implementovat rozhraní pro nastavování vlastností, ukládání, popřípadě vykreslování a bude zobrazen jako uzel v projektovém stromu se svojí ikonou, která bude určovat typ daného souboru.

4.3 Komponentové třídy

Komponentovými třídami chápeme jednotlivé objekty, které jsou určeny pro vkládání do wysiwyg editoru, popřípadě ty, ze kterých se skládá výsledná stránka. Mezi navrženými komponentami bude základ stránky, tlačítka, textová pole, popisky a další prvky, které se užívají k vytváření webových stránek. Protože budeme vytvářet převážně formuláře, jejichž rozložení se většinou udržuje pomocí tabulky, tak nesmíme zapomenout dodat důležitou komponentu a to tabulku, která má mnohé vlastnosti a může mít mnoho různých tvarů.

V dnešním trendu se směr programování orientuje na skládání z komponent. Známé aplikace, například Word, jsou velkými systémy, které se neskládají pouze z jednoho souboru. Když se podíváme na různé takové aplikace, vidíme, že jsou složeny z podobných prvků. Ty jsou ovšem stejné nebo jsou trochu odlišeny. Autor takových aplikací může využívat knihovny volně dostupných komponent na internetu. Šetří si tím čas na svůj projekt.

Proč takový způsob nelze využít v rámci internetových aplikací? Samozřejmě, že lze použít. Ovšem k tomu musí být i uzpůsobená aplikace a WebEditor je právě takovou aplikací. Pomocí wysiwyg editoru budeme vytvářet vzhled webové stránky pomocí komponent, které máme v nabídce. Tyto komponenty můžeme rozdělit na funkční, kontejnerové a pomocné.

Funkčními prvky jsou konečné prvky, které mají svůj vzhled a funkci, například tlačítko, textové pole a další. Kontejnerové prvky mohou v rámci svého vzhledu obsahovat další komponenty.

Mezi takové patří například tabulka, kde obsahem každé buňky musí být další objekt, tím může být i další kontejnerový prvek. Samozřejmě, že celá stránka (nebo spíš objekt ji reprezentující), je kontejnerovým prvkem. Obsahuje totiž všechny prvky, které jsou umístěny na stránce, popřípadě jednu tabulku, pokud má uživatel možnost si zvolit pozicování dle tabulky nebo absolutně. Pomocnými objekty chápeme prvky, které nemají svůj specifický vzhled, ale slouží pomocí svých vlastností k nastavení funkčnosti, popřípadě vzhledu celé stránky. Objekt stránky je takovým prvkem. Svou vlastní grafickou reprezentaci nemá, ale nabízí základní vlastnosti pro nastavení stránky a vrací celý text pro její kódovou reprezentaci.

Z uvedených podkapitol vyplývá, že model bude opravdu složitý a hodně rozvětvený. Budou se využívat abstraktní třídy, rozhraní a dědičnost. Jak dopadl návrh je uvedeno v následujících kapitolách a v příloze ve výsledném diagramu tříd.

5 Návrh řešení

Při návrhu modelu řešení jsem postupoval podle jednotlivých částí, které jsou uvedeny ve čtvrté kapitole. Model tříd sem několikrát pozměnil k jednodušší variantě, které vznikaly postupnou implementací a rozkladu problémů na menší díly. Velkým problémem pro mě byl wysiwyg editor, který byl od samého počátku velkou neznámou. Neměl jsem představu konkrétní implementace v Javě a tak jsem musel Javu studovat a hledat další zdroje, které by mi pomohli. Proto se model tříd několikrát měnil a může obsahovat mrtvé fragmenty.

Při postupném implementování tříd a rozhraní se vytvářela širší funkčnost celé výsledné aplikace a zajišťovaly se určité detaily pro další přidávání. Jednotlivé krůčky určovaly další drobné změny modelu. Nyní si probereme jednotlivé části modelu.

5.1 Použité návrhové vzory

V průběhu řešení modelu a implementace jsem se rozhodnul pro využití některých návrhových vzorů. Nejdříve bych chtěl říci co to jsou návrhové vzory.

5.1.1 Návrhové vzory

Při návrhu různých informačních systémů a dalšího software jsem se setkal s částmi zdrojových souborů nebo lépe řečeno s částmi modelového řešení, které jsem neustále využíval a implementoval do všech svých projektů. Zjednodušeně řečeno jsem užíval své standardní postupy pro řešení problémů. Ano, užíval jsem určitou šablonu, vzor, který se neustále opakoval.

Při studiu zdrojů na diplomovou práci jsem zabrousil na fórum, kde se řešil můj problém a jedna odpověď z mnoha obsahoval text: „Mrkni se na návrhové vzory, vždyť je to singleton“. Načež jsem se dal do hledání a našel jsem zajímavosti.

Návrhové vzory představují moderní téma mezi objektově orientovanou informační společností. Zvláště tou částí, která se zabývá vývojem a návrhem softwaru. Návrhové vzory lze popsat jako obecný popis stále se opakujících a vyskytujících problémů. Tyto vzory se neustále postupně rozvíjí s postupem dokonalejších řešení daných problémů. Vymezení návrhových vzorů jako popisu řešení je velmi strohé. Kdo jiný má možnost tyto návrhové vzory zdokonalovat a dále řešit, než analytici a návrháři, kteří řeší netriviální a stále se opakující stejné problémy.

Návrhové vzory můžeme rozdělit do dvou skupin:

Implicitní – nepopsaná řešení, která vznikla ze znalostí a zkušeností

Explicitní – zdokumentované zkušenosti při řešeních konkrétních problémů.

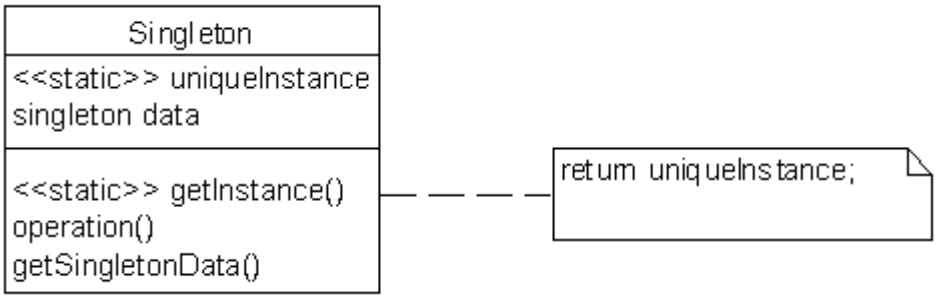
Explicitních návrhových vzorů je mnohem méně. Pro vytvoření návrhového vzoru, tedy jeho popis, se vyplatí tehdy, je-li výskyt řešeného problému návrhovým vzorem velmi častý. Pokud tato podmínka neplatí, neoplatí se dokumentovat dané řešení.

Mnohé návrhové vzory vyšly z různých programovacích jazyků. Tedy můžeme říct, že návrhový vzor není osamocenou jednotkou, ale výsledek kombinace praktických a teoretických zkušeností. Proto by měl programátor znát velice dobře prostředí, ve kterém pracuje, aby mohl tyto vzory aplikovat a hlavně, věděl jak.

Zde jsem z počátku narazil na první problémy neznalosti platformy Javy, a proto některá mnou navržená řešení nejsou optimální.

5.1.2 Singleton

Asi mnou nejvíce používaný návrhový vzor je Singleton, česky Jedináček. Singleton patří mezi takzvané tvořivé návrhové vzory. Tím se myslí, že vytváří nějakou instanci objektu, který tento Singleton reprezentuje. Pokud potřebujeme změnit chování, dosáhneme toho pomocí výměny instance objektu.



Obrázek 7: Model návrhového vzoru Singleton

Tento návrhový vzor se užívá tam, kde je zapotřebí zajistit pouze jednu jedinou instanci dané třídy a zpřístupnit ji širokému okolí, to znamená je globálně přístupná odkudkoliv. Při implementaci tohoto návrhového vzoru se musíme řídit daným programovacím jazykem a zajistit správnost vytváření dané instance.

Řešení Singletonu spočívá ve vytvoření třídy, která má všechny konstruktory s viditelností private. Obsahuje jeden statický atribut `uniqueInstance`, který obsahuje vytvořenou instanci a pomocí statické metody `getInstance()` si může kdokoliv vzít referenci na danou instanci a pracovat s objektem jak je mu libo. Třeba použít metodu `getSingletonData()`. Příklad implementace Singletonu v Javě vypadá takto:

```
public class Singleton {
     private static Singleton uniqueInstance;
     private Singleton() {
     }
```

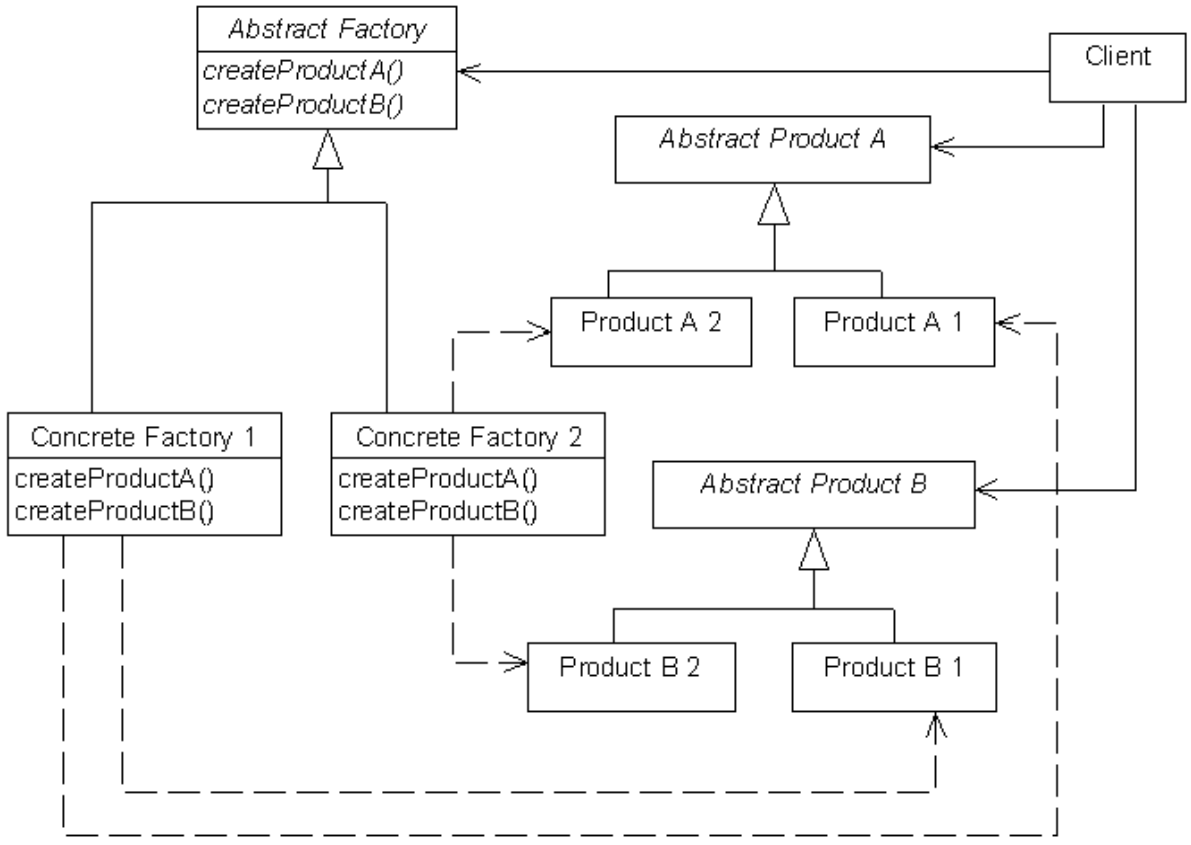
```
        public static synchronized Singleton getInstance() {
            if (uniqueInstance == null) {
            uniqueInstance = new Singleton();
        }
        return uniqueInstance;
    }
    public void operation() {
        System.out.println("Singleton.operation() executing" );
    }
}
```

Potom již příklad použití je jasný a jednoduchý:

```
Singleton.getInstance().operation();
```

V metodě `getInstance()` se nejdříve zkontroluje, zda je instance vytvořena a pokud není, tak se vytvoří a vrátí se její reference.

5.1.3 Abstract Factory



Obrázek 8: Model návrhového vzoru Abstract Faktory

Tento návrhový vzor opět patří mezi vzory tvořivé. A již víme od vzoru Singleton, že pro změnu processu tvorby se užije změna instance továrny. Abstract Factory zaštiťuje více tříd, které vytvářejí

instance konkrétních produktů. Tyto třídy vytvářejí různé instance (produkty), které ovšem spolu nějak souvisí. Přejdeme ihned k popisu a třídnímu modelu tohoto vzoru.

Na modelu vidíme třídu `Abstract Factory`, která implementuje dvě abstraktní metody pro získání reference jednotlivých produktů. Těmi produkty jsou rozhraní `Abstract Product A` a `Abstract Product B`. Klient má umožněn přístup k abstraktní továrně a k abstraktním produktům.

Při pokusu o získání instance z abstraktní továrny se vytvoření instance konkrétní továrny, která implementuje metody pro získání daných rozhraní z konkrétních tříd produktů, které jsou určené pro danou továrnu. Při získávání instance továrny se využívá návrhového vzoru Singleton, který vytvoří jedinou instanci dané konkrétní továrny.

Předběžně prozradím, že tento návrhový vzor využívám pro vytvoření databázové aplikační vrstvy.

5.2 MySQL rozbor

Aplikace má pracovat s databází a podle ní generovat formulář a kód. Pro správné vygenerování kódu a formuláře vůbec je potřeba se podívat i na stranu databázovou a zjistit informace z této strany. Jak jsem již zmínil dříve, pro práci s databází nám bude sloužit implementace návrhového vzoru Abstract Factory a díky tomuto vzoru si vytvoříme později dané produkty.

Které základní informace potřebujeme získat z databáze, a které od uživatele? Od uživatele to bude minimální počet informací. Celkem to jsou čtyři položky:

- Server
- Uživatelské jméno
- Heslo
- Schéma

Položka server nám určuje adresu počítače, kde databázový stroj běží. Pro přístup k databázovému stroji se potřebujeme přihlásit a k tomu slouží položky uživatelské jméno a heslo. Toto jsou všechny nejpotřebnější informace, protože bez názvu schémata se můžeme bez problému přihlásit. Potom musím ale zadat příkaz pro výběr správného schématu. Schématem se zde myslí jedna interní databáze tabulek v MySQL serveru. Tímto termínem se snažím rozlišit databázový server od souboru tabulek.

Jak zjistíme po přihlášení k databázi, které schémata máme k dispozici?

5.2.1 MySQL dostupná schémata

MySQL databázový server má vlastní interní bezpečnostní systém, kde lze nastavit uživatelům databáze přístupy k jednotlivým položkám v databázi. Těmi položkami jsou schémata, tabulky, sloupce, pohledy a dále můžeme specifikovat jednotlivé přístupy a akce.

Pro zisk všech dostupných schémat, dle nastavených zabezpečovacích pravidel, nám poslouží jednoduchý příkaz:

```
show databases;
```

Tento příkaz nám vrátí jako odpověd následující tabulku:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
| webeditor          |
+--------------------+
```

Z této odpovědi si potom můžeme vybrat schéma, se kterým chceme dále pracovat. Jak můžeme vidět v názvu tabulky se místo schéma užívá databáze. Pokud ovšem budete užívat nástroje od MySQL pro správu MySQL databáze, tak se setkáte s pojmem schéma. Nyní už víme informace o tabulkách, a které další potřebujeme? Sloupce.

5.2.2 Schéma information_schema

Zmiňoval jsem, že se podíváme na sloupce a najednou je tady nějaké schéma `information_schema`, co sním? To řeknu později, nejdříve jdeme k sloupcům.

Jak víme, tabulka obsahuje sloupce. Každý sloupec je definován nějakými hodnotami. Tak si je pěkně probereme. Sloupec do tabulky můžeme přidat při vytváření tabulky pomoci příkazu, kde část, která se týká definice sloupce je následující:

```
column_definition: col_name data_type [NOT NULL | NULL]
                   [DEFAULT default_value] [AUTO_INCREMENT]
                   [UNIQUE [KEY] | [PRIMARY] KEY]
                   [COMMENT 'string'] [reference_definition]
```

Položka data_type specifikuje datový typ sloupce a na ně se podívám podrobněji dále. Všechny ostatní hodnoty musí databáze někde uložit a k tomu slouží právě schéma `information_schema` zvláště tabulka `columns` . Pro nás budou nejdůležitější položky z tabulky `columns`:

```
COLUMN_NAME
TABLE_NAME
COLUMN_DEFAULT
IS_NULLABLE
DATA_TYPE
CHARACTERT_MAXIMUM_LENGTH
COLUMN_KEY
EXTRA
```

Nyní víme, které položky budeme potřebovat a kde je najdeme. Jen poslední položku jsem ještě nezmínil a tím jsou datové typy. Víme, že hodnotu najde v tabulce `columns`, ale co vše najdeme?

5.2.3 MySQL datové typy

Seznam všech datových typů si vypisovat nebudeme. Budu se soustředit pouze na ty, které jsem se snažil naimplementovat, ale i přesto je jich dosti.

5.2.3.1 Číselné typy

MySQL má podporu třech druhů číselných formátů. Jsou jimi celočíselné, reálné a přesné německé datové typy:

- Celočíselné
  - `TINYINT (1B)`
  - `SMALLINT (2B)`
  - `MEDIUMINT (3B)`
  - `INT (4B)`
  - `BIGINT (8B)`
- S desetinnou čárkou
  - `FLOAT`
  - `DOUBLE`
- Přesné číslo
  - `DECIMAL`

Samozřejmostí je, že můžeme nastavit příznaky `SIGNED|UNSIGNED`, popřípadě počet číslic. Ve své aplikaci jsem se rozhodl, že pro číselné typy budu generovat komponentu typu textové pole. Dalším možným rozšířením aplikace je aplikace kontrolních mechanismů pro jednotlivé číselné vstupy.

5.2.3.2 Řetězcové typy

Pro uložení textových vstupů si můžeme vybrat ze třech možností. Řetězcový typ `CHAR` slouží k ukládání řetězců konstantní délky. Typ `CHAR` má maximální rozsah 0 – 255 znaků a při vytváření sloupce můžeme určit, kolik znaků se bude ukládat, neboli maximální délku řetězce. Řetězec je u tohoto typu vždy uložen v plné délce. Pokud se pokusíme uložit kratší řetězec, než jsme definovali délku při vytváření, bude řetězec doplněn mezerami do plné délky.

Přesným opakem je typ `VARCHAR`, který při ukládání kratších řetězců, než je definován daný sloupec nic nedoplní a uloží řetězec na co nejmenším paměťovém prostoru. `VARCHAR` má maximální rozsah 0 – 255 znaků, ale toto platí pouze do verze MySQL 5.0.3, protože od verze MySQL 5.0.3 a výše je tento rozsah zvětšen na 0 – 65535 znaků.

Posledním typem, do kterého můžeme ukládat i velice dlouhé řetězce je `TEXT`. Jedná se vlastně o pole znaků a rozsah tohoto typu je opravdu rozsáhlý. Typ `TEXT` má čtyři další podtypy, které se liší pouze maximální délkou, kterou jsou schopny uložit. Jsou to:

- `TINYTEXT` $< 2^8$
- `TEXT` $< 2^{16}$
- `MEDIUMTEXT` $< 2^{24}$
- `LONGTEXT` $< 2^{32}$

Pro textový typ `TEXT` jsem zvolil vstupní komponentou element typu `TextArea`. Pro ostatní typy `CHAR` a `VARCHAR` je to textové pole.

5.2.3.3 Výčtový typ

Jedná se o klasický výčet hodnot, kde do daného sloupce lze uložit pouze jednu jedinou hodnotu z tohoto výčtu. I když máme výčet definovaný například takto:

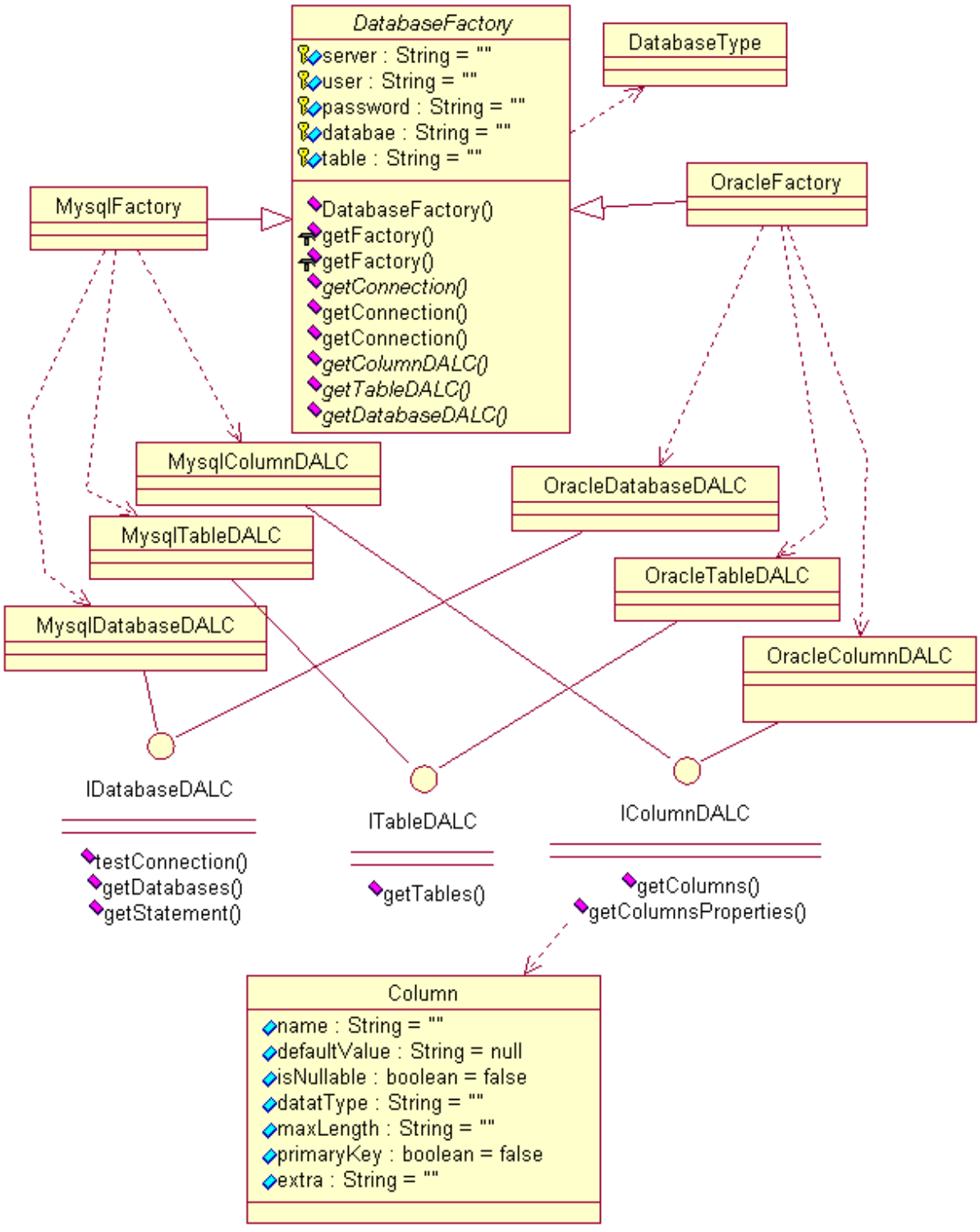
```
ENUM ('one', 'two', 'three'),
```

Máme k dispozici následující hodnoty, které můžeme využít pro uložení do sloupce:

```
NULL = NULL
'' = 0
'one' = 1
'two' = 2
'three' = 3
```

Pro tento datový typ jsem zvolil element select, který odpovídá rozbalovacímu seznamu, ze kterého lze vybrat právě jednu z možných hodnot.

6 Model řešení a implementace



Obrázek 9: Implementovaný návrhový vzor Abstract Factory

Model řešení jsem provedl pomocí aplikace RationalRose a implementaci jsem provedl v již zmíněném programovacím jazyce Java ve vývojovém prostředí NetBeans. Model řešení můžeme rozčlenit na samostné bloky, které si postupně ho zde projdeme. První částí, kterou si projdeme, bude napojení na databázi.

6.1 Navržený model Abstract Factory v praxi

Již v předchozí kapitole byly zmíněné návrhové vzory, zejména Abstract Factory. Tato část modelu je uzavřena do svého vlastního balíčku `webeditor.database`. Model `DatabaseFactory` můžete najít na předchozí stránce. Tento model je rozšířen o třídy, které vytvářejí implementaci konkrétních produktů pro databázi Oracle. V aplikaci jsou pouze implementovány produkty pro MySQL. Oracle databáze je zde vyobrazena pro přehled, jak lehce stačí rozšířit celou aplikaci o podporu nové databáze.

Abstraktní třída `DatabaseFactory` implementuje přetíženou metodu `getFactory()`, která vrací instanci konkrétní továrny. Druhá implementace této metody má vstupním parametrem typ databáze `DatabaseType` pro možnost vytvoření instance konkrétní továrny daného typu. Samozřejmě, že nově vytvořenou instanci ihned vrátí. Právě tyto dvě metody implementují návrhový vzor Singletone.

Při pokusu o získání spojení s databází pomocí metody `getConection()`, se vložené atributy ihned uloží a poté už můžeme pouze požadovat spojení s databází pomocí metody bez parametrů. Tohoto využívají všechny metody, které jsou implementovány v konkrétních třídách produktů. Nemusí se dotazovat na jednotlivé parametry pro vytvoření spojení s databází. Každé nové volání metody `getConection()` vrací novou instanci třídy `Conection`.

Jako produkty abstraktní továrny jsou zde rozhraní, která nabízejí metody pro práci s databází, myšleno jako se schématem, s tabulkou a sloupci v dané tabulce. Jsou zde metody `getDatabases()`, `getTables()` a `getColumns()`, které vracejí pouze jména dostupných položek pro výběry uživatele.

Pro získání podrobností o sloupcích se využívá třída `Column`, která shromažduje dříve vyčtené položky z tabulky `Columns` ze schématu `information_scheme`. Tyto položky jsou využity ve třídě `WebFile` pro automatizované vytváření formuláře.

6.2 Vlastnosti komponent

Všechny komponenty, které aplikace nabízí k práci s wysiwyg editorem, obsahují řadu vlastností. Proto jsem pro rychlou a bezproblémovou práci s nimi naimplementoval efektivní nástroj. Programovací jazyk Java nabízí několik implementací nástrojů pro udržování různých atributů, avšak žádný mi dostatečně nevyhovoval.

Základem mého nástroje pro práci s vlastnostmi je třída `Property`, která obsahuje základní metody pro získání a nastavení hodnot dané vlastnosti. Daná vlastnost má následující položky:

- `name` – identifikátor vlastnosti
- `title` – titulek pro okno vlastností
- `value` – hodnota vlastnosti
- `type` – typ vlastnosti
- `readOnly` – určuje vlastnost pouze pro čtení
- `viewAble` – určuje viditelnost vlastnosti

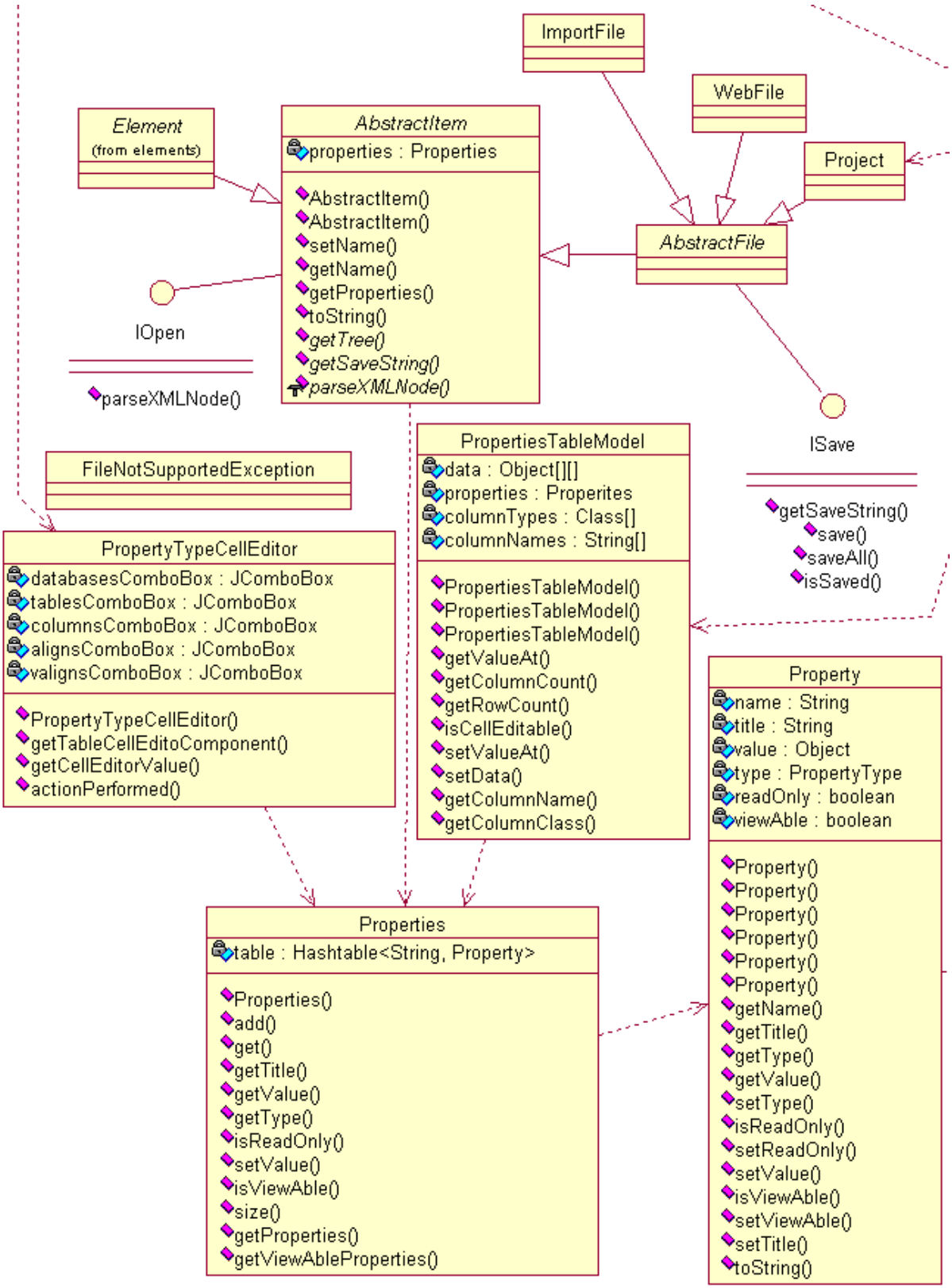
Všechny tyto položky jsou potřebné pro bezproblémovou práci s tabulkou vlastností, kde potřebujeme vědět více informací než je právě jen hodnota. Položka `viewAble` má za následek zobrazení, popřípadě nezobrazení dané vlastnosti. Položka `readOnly` určuje editovatelnost položky v seznamu.

Žádná komponenta nemá pouze jednu vlastnost, a tak třída `Properties` implementuje jednoduchý kontejner pro tyto vlastnosti a nabízí základní metody pro přímé nastavování a získávání hodnot z vlastnosti. Všechny tyto informace zpracovává a aplikuje třída `ProperiesTabelModel`, která je odvozená od `DefaultTabelModel` a přetěžuje základní ovládací metody.

Položka `type` je výčtového typu `PropertyType` a určuje typ dané vlastnosti, který je využíván třídou `PropertyTypeCellEditor`. Již ze samotného názvu lze usoudit, že tato třída bude poskytovat různé editující komponenty pro úpravu hodnot vlastností. Takovýmto způsobem jsem docílil podobného chování, které můžeme vidět v profesionálních vývojových prostředích.

6.3 Projektový strom

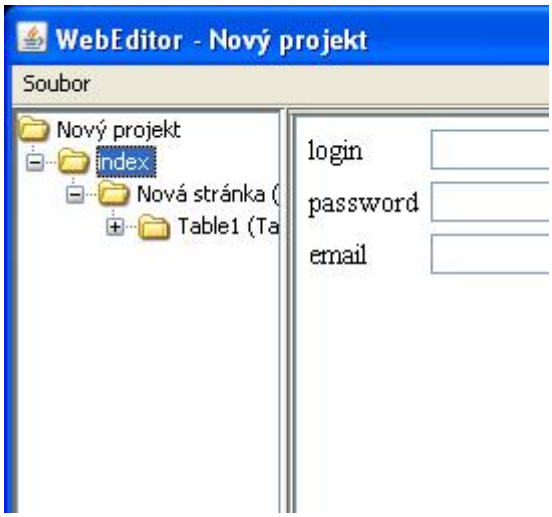
V předchozí kapitole jsem probíral vlastnosti a na následující stránce je vidět model tohoto nástroje. Můžeme si všimnout, že hlavní třída `Properties` je součástí abstraktní třídy `AbstractItem` a z upraveného modelu vidíme, že `AbstractItem` je společným předchůdcem všech položek a komponent v aplikaci.



Obrázek 10: Implementace nástroje pro správu vlastností

Díky společnému předchůdci můžeme v projektovém stromě zobrazovat všechny přítomné komponenty a soubory. Tímto způsobem implementace jsem docílil podobného chování, které můžeme vidět v různých vývojových nástrojích.

Pomocí třídy `WebEditorTreeCellRenderer` můžeme ovlivnit vykreslování ikon k jednotlivým položkám daného stromu. Příkladem může být ikona projektu, která se nikdy nemění v závislosti na tom, zda obsahuje, či neobsahuje potomka. Další využití této možnosti jsem neimplementoval, poněvadž jsem se zabýval důležitější částí, a to wysivyg editorem.



Obrázek 11: Ikona projektu s obsahem

Obrázek 12: Ikona samotného projektu

6.4 Implementace komponent

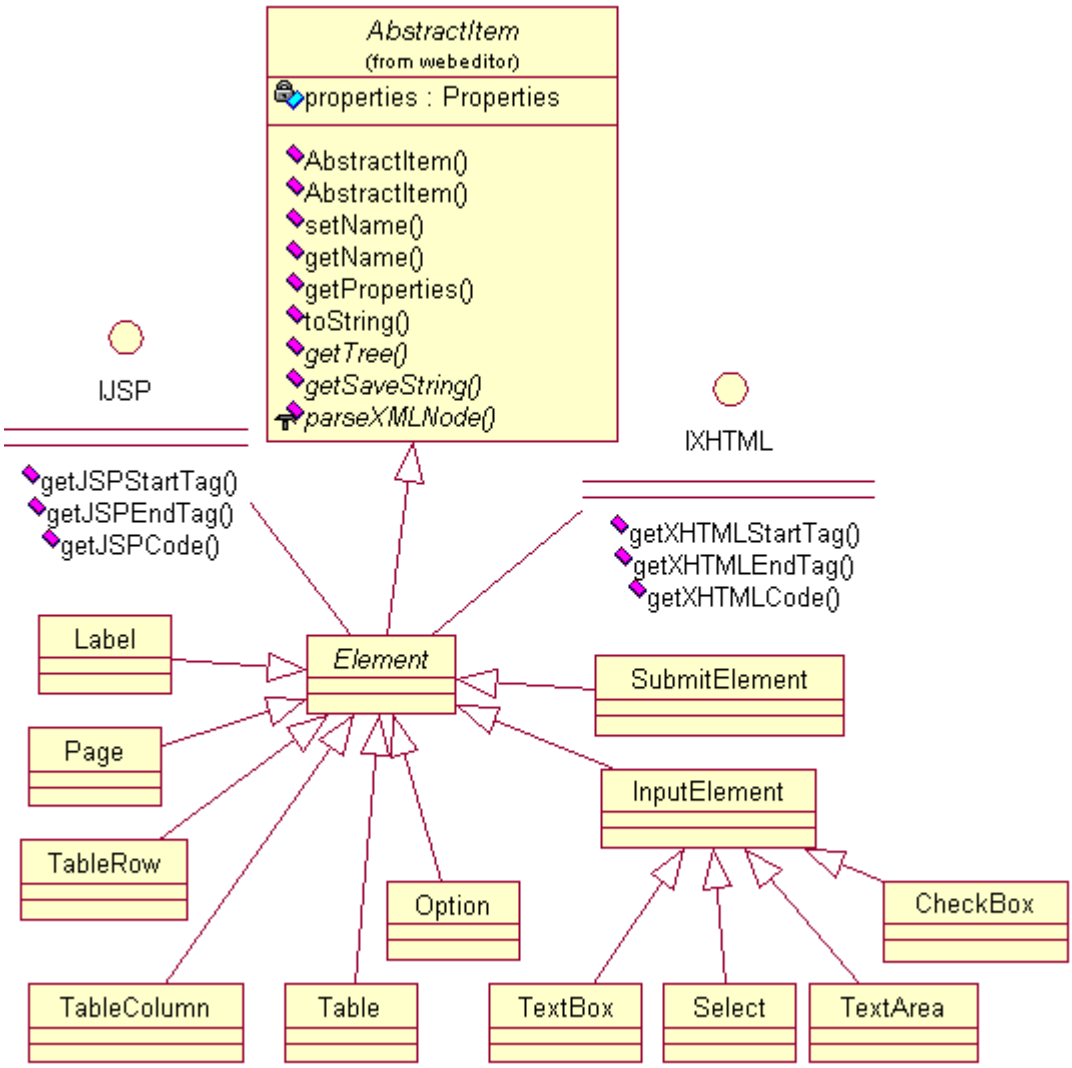
Komponenty pro práci s wysivyg editorem jsem uložil do balíku s názvem `webeditor.elements`. Základní třídou pro všechny komponenty je abstraktní třída `Element`, která dědí od společné kořenové třídy `AbstractItem` a implementuje rozhraní `IXHTML` a `IJSP`.

Tato rozhraní obsahují metody nutné ke generování výsledného JSP kódu, `IJSP` rozhraní. Druhé rozhraní se zdá být nadbytečné, jelikož uživatel nikdy tento vygenerovaný kód neuvidí, ale přesto je velice zapotřebí, protože je důležitý pro vytváření vzhledu, který nám zobrazuje wysivyg editor.

Další významnou abstraktní třídou je `InputElement`, od kterého se dědí všechny vstupní komponenty. Výhodou takovéto společné třídy je snadná detekce a lehké hledání ve stromu komponent. Hledání vstupních elementů probíhá pomocí identifikace příslušnosti k dané třídě. Vyhledávání všech vstupních komponent je důležité pro generování JSP skriptu, protože v rámci generování potřebujeme mít všechny prvky lineárně a rychle přístupné oproti neustálému prohledávání celého stromu.

Stejné chování využívám u třídy `SubmitElement`, od kterého by měly být odvozeny všechny aktivní komponenty, které můžou způsobit odeslání formuláře na server. Tato třída definuje vlastnost `typeAction`, která určuje chování těchto aktivních elementů, obzvlášt tlačítka submit.

Vlastnost `typeAction` může nabývat hodnot insert, update a delete, které definují generování skriptu do skripletové části JSP kódu.



Obrázek 13: Implementace komponent pro wysiwyg editor

6.5 Generování formuláře

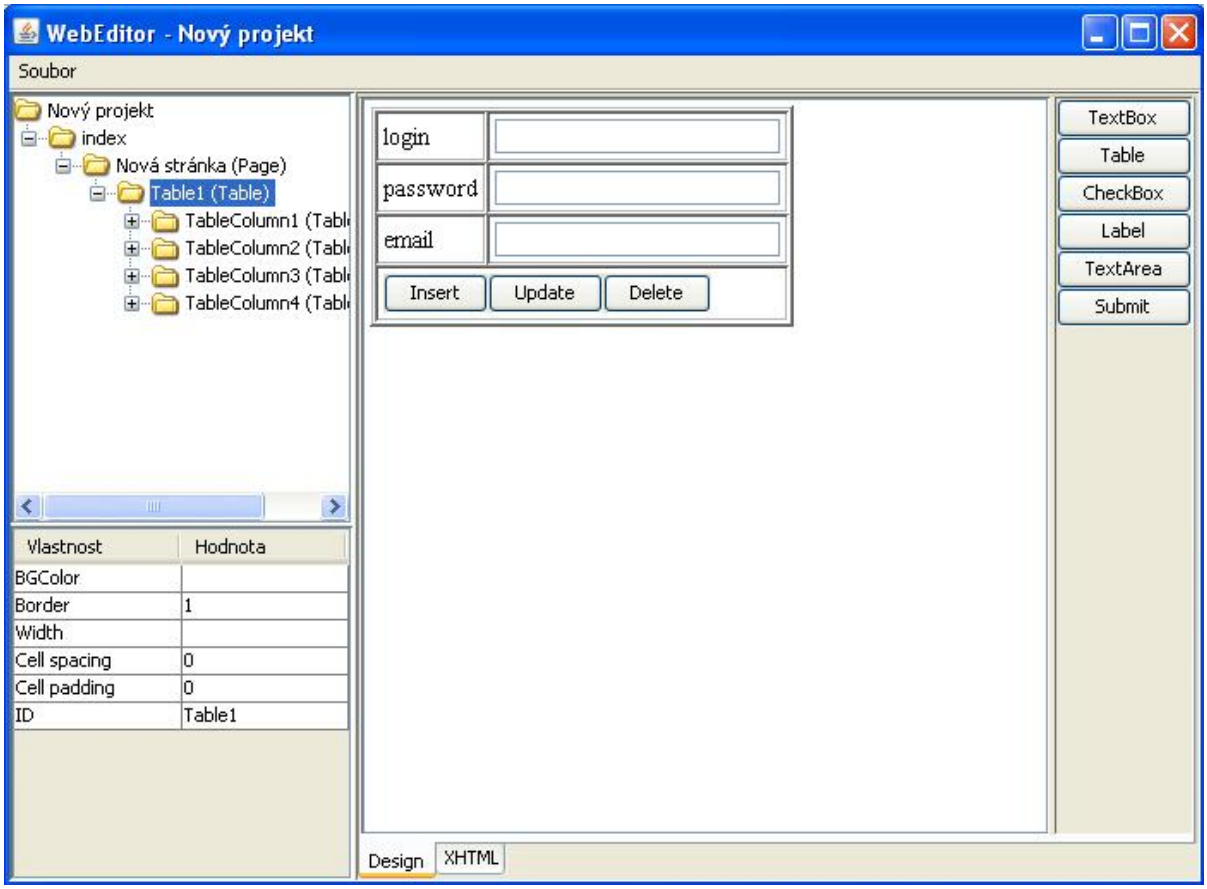
Generování formuláře probíhá automaticky při vytvoření nového souboru JSP. Uživatel je nucen vybrat tabulku, pokud chce vytvořit tento typ souboru. Vytvoří se objekt typu `WebFile`, který je potomkem abstraktní třídy `AbstractFile` a implementuje metodu `generateForm`().

Tato metoda pomocí továrny `DatabaseFactory` získá instanci produktu `IColumnDALC`, kde pomocí metody `getColumnProperties()` získá informace o sloupcích dané tabulky a generuje

dvousloupcovou tabulku. V levém sloupci je obsažen název sloupce, v pravém sloupci je obsažena příslušná vstupní komponenta, která náleží datovému typu sloupce.

V kapitole 5.2.3, kde jsme rozebírali datové typy, jsem se zmínil o příslušnosti jednotlivých datových typů k daným komponentám. Základní komponentou je `TextBox`, který postihuje největší část datových typů. V rámci možných rozšíření této aplikace je vytvoření lepší metody a komponent, které budou hlídat uživatelem vkládaná data.

Na závěr tabulky jsou vygenerována tlačítka pro ovládání formuláře a ke každému tlačítku je přiřazena právě jedna akce. K daným tlačítkům s nastavenou vlastností `typeAction` se zpětně vygeneruje kód v Javě pro daný typ příkazu. Pro zobrazení hodnot na formuláři stačí zadat pomocí metody Get id příslušného záznamu do proměnné, která se nazývá stejně jako sloupec s primárním klíčem databázi.



Obrázek 14: Implementovaná aplikace se zobrazeným wysiwyg editorem

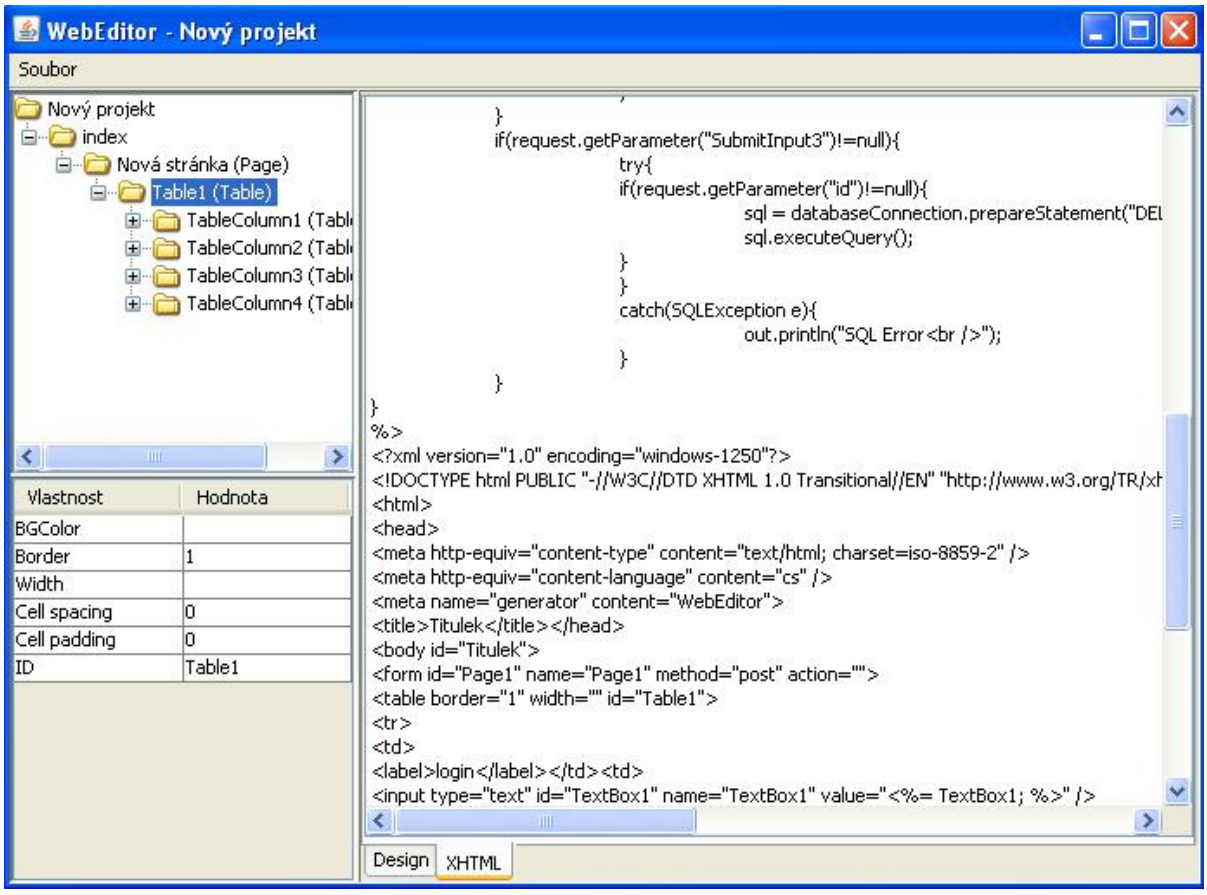
6.6 Implementace wysiwyg editoru

Implementace této části je na tomto projektu nejsložitější. Dlouhou dobu mi trvalo přijít na způsob svázání generovaného textu s událostmi od uživatele pomocí myši. Všechny dostupné komponenty, které nabízí programovací jazyk Java nabízejí různé zobrazovací komponenty pro jednoduché HTML dokumenty. Avšak nevýhodou těchto komponent je velice složitý způsob ovládání a spravování,

obzvláště pokud chceme zjistit kliknutím myši nějaký aktivní prvek HTML dokumentu. Tento prvek je vygenerován knihovními třídami a uživatel se nemůže k těmto vygenerovaným objektům dostat. Snažil jsem se vyřešit tento problém dlouhou dobu a podařilo se mi naimplementovat dynamické zadávání komponent.

Při různých pokusech o implementaci řešení daného problému jsem se dostal do fáze, kdy aplikace šla bez problému přeložit, avšak nechtěla se spustit. Tuto situaci jsem řešil několik dnů a výsledkem byl nucený návrat k předchozí neúplné verzi.

Zadávání funguje bez problému, objekty se vytvářejí a vkládají se do stromové struktury. Pozice, kam se má daná komponenty vložit, se získává z XHTML interního zdroje, který je uložen v HTML dokumentu, který je součástí `wysiwygPane`. Díky tomuto dokumentu získáme aktuální pozici textového kurzoru a pomocí rozborů HTML dokumentů lze najít nejbližší komponentu, která nám určí pozici ve stromové struktuře. Na tuto pozici pak vložíme danou novou komponentu.



Obrázek 15: Implemenovaná aplikace se zdrojovým editorem

6.7 Práce s programem

Na začátku musí uživatel vytvořit nový projekt. K vytvoření slouží dialog, ve kterém je uživatel tázán na název projektu, cestu, uložení, databázový server, uživatelské jméno a heslo. Po zadání všech těchto údajů se povolí tlačítko Test conection, které otestuje připojení k danému serveru a v případě

úspěchu vrátí seznam dostupných schémat. Uživatel musí jedno schéma vybrat a pak potvrdit formulář. Tím je projekt vytvořen.

Jednotlivé stránky se přidávají pomocí nabídky Nový soubor a uživatel je dotázán pouze na typ souboru a název souboru. V případě, že se jedná o JSP soubor, je uživatel ještě dotázán na tabulku databáze. O potvrzení formuláře je automaticky vygenerován formulář, který je obsluhován pomocí třech základních tlačítek: Insert, Delete a Update. Pokud chceme zobrazit ve formuláři nějakou položku databáze, musíme stránce předat identifikátor pomocí metody GET.

Závěr

Aktuální stav implementované aplikace se ještě zdaleka neblíží plánovanému cíli, jelikož cíle byly vzaty z projektů, které se vyvíjejí již několik let, a na kterých pracuje několik týmů. Je zřejmé, že napsání takovéto aplikace s danými funkčními výhodami, jakým wysiwyg editor zcela určitě je, je pro jednoho člověka hodně velké sousto. I po různých peripetiích s vývojovým prostředím jsem dosáhl nemalého cíle.

V současné době bych již tento projekt řešil pomocí softwarových balíků, které již nabízejí základní editory s podporou projektového stromu, s tabulkou vlastností a základních vyspělejších editorů. Pro příklad bych chtěl zmínit NetBeans platform a jistě existují i další.

I přesto si tento projekt zaslouží určitá vylepšení a to nejen ty, které již byly uvedené, ale zvláště věci, které se mi nepodařilo dopracovat do úplného konce. Chtělo by vylepšit práci s dokumentem wysiwyg editoru, dodat zvýraznění syntaxe, rozšířit nabídku komponent a v neposlední řadě doladit grafickou stránku aplikace.

Při řešení mě napadlo několik zadání dalších prací, které by se týkali vytvoření podobné aplikace, avšak práce by byly rozděleny mezi více vývojářů a bral by si za cíl vytvořit univerzální editor s lehkým způsobem dalšího rozšiřování na specifické případy.

Literatura

[1] BRANICKÝ, Marek. Java Servlets - predstavenie technológie [online]. 25. 4. 2003 [cit. 2007-05-22]. Dostupný z WWW: <http://interval.cz/clanky/java-servlets-predstavenie-technologie/>. ISSN 1212-8651.

[2] BRANICKÝ, Marek. JavaServer Pages pro všechny [online]. 6. 8. 2002 [cit. 2007-05-22]. Dostupný z WWW: <http://interval.cz/clanky/javaserver-pages-pro-vsechny/>. ISSN 1212-8651.

[3] Hawlitzek, F.: Java 2 příručka programátora, Praha, Grada, 2002

[4] Herout, P.: Učebnice jazyka Java, České Budějovice, Kopp, 2003

[5] Hall, M.: Java Servlety a stránky JSP, Praha, Neocortex s. r. o., 2001

[6] Fórum k Javě: http://forum.builder.cz/list.php?14 [10. 5. 2005]

[7] Návody na Javu: http://java.sun.com/docs/books/tutorial/search.html [10. 5. 2005]

[8] Dokumentace J2SE: http://java.sun.com/j2se/1.5.0/docs/api/index.html [10. 5. 2005]

Seznam příloh

Příloha č. 1