TECHNOLOGIE MODEL-VIEW-CONTROLLER A JEJÍ UPLATNĚNÍ PŘI NÁVRHU A IMPLEMENTACI INFORMAČNÍCH SYSTÉMŮ V PROSTŘEDÍ TENKÉHO KLIENTA

Ivo Martiník

Ekonomická fakulta VŠB-TU Ostrava, Sokolská třída 33, 701 21 Ostrava 1, ČR
ivo.martinik@vsb.cz

Abstrakt
Příspěvek se zabývá principy a uplatněním principu Model-View-Controller (MVC) při návrhu a implementaci informačních systémů provozovaných v rámci Internetu a Intranetu realizovaných technologiemi J2EE v prostředí tenkého klienta. Technologie MVC je s úspěchem využívána především v rámci implementace aplikační vrstvy informačního systému na straně aplikačního serveru, méně obvyklé je její použití při návrhu a implementaci prezentační vrstvy na straně tenkého klienta (WWW prohlížeče). Příspěvek se zabývá jedním z možných přístupů k řešení uvedeného problému a jeho zajímavým důsledkům při návrhu aplikační vrstvy informačního systému.

1. Úvod

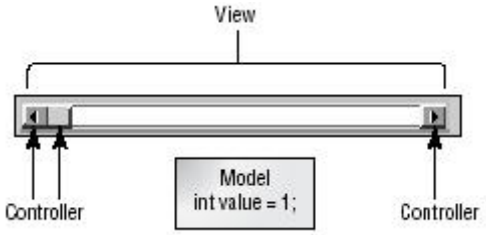
Jednou z významných technologií, jež se již mnoho let výrazně uplatňují při návrhu a implementaci nejen informačních systémů, představuje *Model-View-Controller* (MVC). Koncept MVC je od jeho samotných počátků svázán s paradigmatem objektově-orientovaného programování, technologie bylo poprvé využito v knihovnách tříd programovacího jazyka Smalltalk (viz. např. [1]), její výskyt je typický při programování uživatelského interface a značnou popularitu si rovněž získala v knihovnách tříd Swing programovacího jazyka Java.

Základním rysem MVC technologie je koncept implementace dané programové komponenty (např. nejtypičtěji u grafického uživatelského rozhraní - položka menu, lišta, tlačítko apod.) prostřednictvím tří instancí tříd:

- *View* – objekt implementující vizuální reprezentaci grafické komponenty.
- *Model* – objekt implementující datové položky grafické komponenty.
- *Controller* – objekt implementující reakce na události grafické komponenty, na základě události provádí objekt *controller* akce (volání metod, změny hodnot datových položek) nad objekty *model* a *view*, současně garantuje vazbu mezi hodnotami datových položek objektu *model* a jejich grafickou reprezentací objektem *view*.

Příkladem použití MVC může být např. grafická komponenta scrollbar (viz. obr. 1). Při výskytu události (např. přesunu tlačítka scrollbaru pomocí myši) je vyvolána příslušná událost, na niž reaguje *controller*. Ten zabezpečí příslušnou změnu datové položky objektu *model* a grafické reprezentace objektu *view* (např. rovněž vygenerováním události jistého typu). Při návrhu uživatelského rozhraní programu s využitím MVC je rovněž častým jevem sdílení objektu reprezentujícího *model* několika grafickými komponentami uživatelského rozhraní současně – např. tatáž data jsou na ploše současně reprezentovány textovou (tabulka) a grafickou (graf funkce) formou. Při změně hodnoty dat (např. uživatel edituje

hodnotu jisté položky v tabulce) zabezpečí *controller* změnu příslušné hodnoty ve sdíleném *modelu* a všech reprezentacích modelu prostřednictvím *views*.



Obr. 1: Model-View-Controller

2. Technologie MVC a její uplatnění ve vícevrstvých softwarových architekturách

Při návrhu a implementaci informačních systémů s využitím technologie tenkého klienta je charakteristické využití vícevrstvé architektury, typické je nasazení tří, příp. čtyř softwarových vrstev, jež mohou mít např. následující strukturu:

- *Prezentační vrstva* – je implementována na straně tenkého klienta, jímž je typicky WWW prohlížeč, časté bývá využití technologií HTML, XML, JavaScript, CSS, DOM apod.
- *Web vrstva* – standardně je využita k implementaci prezentační logiky, bývají nasazeny technologie Java Servlets, Java Server Pages, .Net apod.
- *Aplikační vrstva* – implementuje aplikační logiku v prostředí aplikačního serveru, technologie Enterprise Java Beans, .Net, CORBA apod.
- *Databázová vrstva* – je reprezentována vybraným RDBMS – Oracle, IBM DB2, MS SQL, MySQL apod.

Technologie MVC lze úspěšně využít při návrhu programových systémů s vícevrstvou architekturou. Jednotlivé vrstvy jsou pak reprezentovány následovně:

- *Prezentační vrstva* – na straně tenkého klienta
- *Web vrstva* – implementuje prezentační logiku, reprezentuje komponentu *view* na straně serveru, při využití J2EE technologie (viz. [2]) je typicky implementována technologií Java Server Pages (JSP)
- *Aplikační vrstva* – implementuje aplikační logiku, reprezentuje komponentu *controller* na straně serveru, využití technologií Java Servlets, Enterprise Java Beans, .Net
- *Databázová vrstva* – reprezentuje komponentu *model* na straně serveru, technologie RDBMS

Výrazné uplatnění této filozofie lze mj. nalézt v rámci produktu Struts (viz. [3]), jež je organickou součástí projektu Jakarta (viz. [4]). Uvedený produkt je věnován specificky řešení problému návrhu a implementace Web aplikací s podporou nasazení technologií Java Servlets (v roli komponent *controller*) a Java Server Pages (v roli komponent *view*) na straně aplikačního serveru, přičemž přístup ke komponentám *model* je standardně realizován rozhraním JDBC.

3. Implementace prezentační vrstvy programového systému na straně tenkého klienta s využitím technologie MVC

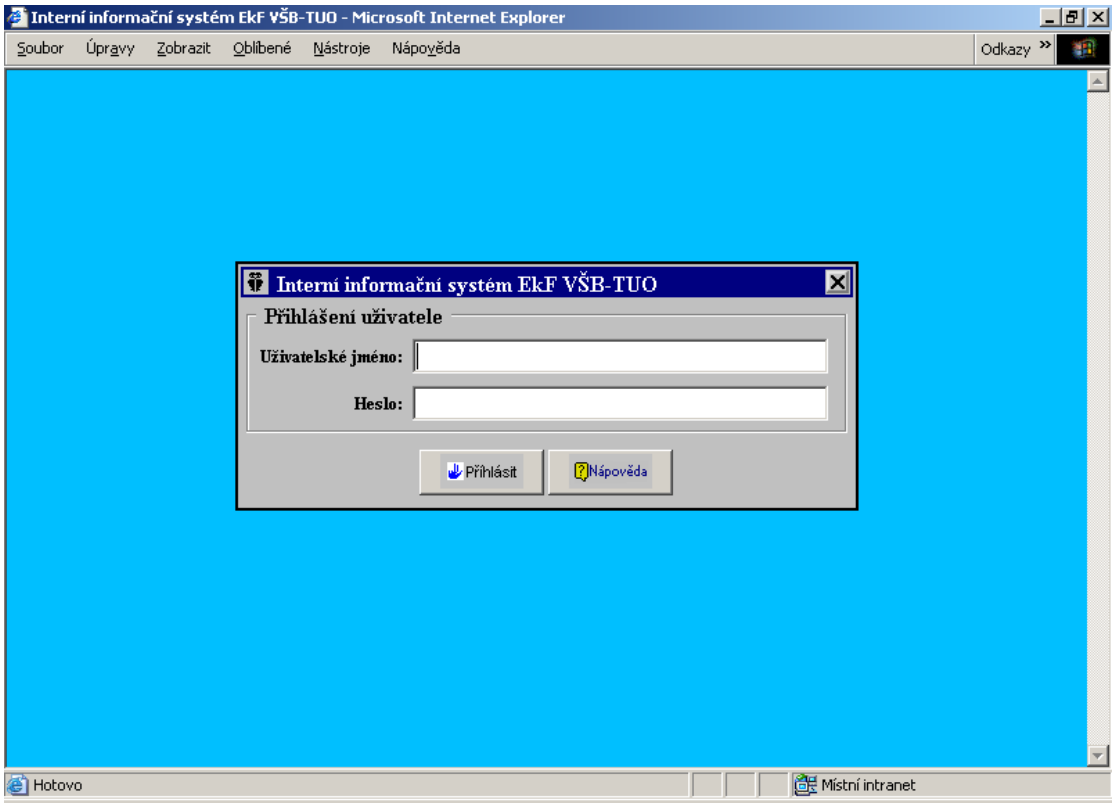
V současné době je na pracovišti autora příspěvku realizován projekt nové verze informačního systému fakulty, pro jehož implementaci byly řešitelským kolektivem mj. stanoveny následující požadavky:

- Implementace bude nezávislá na hardwarové a softwarové platformě strany klienta i serveru.
- Veškerý uživatelský interface bude dostupný v prostředí WWW prohlížeče bez nutnosti instalace přídavných plug-in komponent (zejména Java Virtual Machine, Flash apod.) s dosažením maximálního komfortu uživatelského interface.
- Prostředí informačního systému bude integrováno s jinými systémy využívanými pedagogy a studenty fakulty (zejména pak s prostředím řídicího systému virtuální univerzity, jímž je v současné době produkt Tutor2000 firmy Kontis s.r.o. – viz [5]).

Řešitelský kolektiv se proto rozhodl pro nasazení následujících technologií:

- *Prezentační vrstva* řešení bude implementována s využitím technologií HTML, JavaScript, Document Object Model (DOM – viz. [6]) a Cascading Style Sheets (CSS – viz. [7]). Vzhledem k rozdílným implementacím těchto technologií v prostředí jednotlivých WWW prohlížečů (např. MS Internet Explorer v. 6 implementuje CSS Level 1 a DOM Level 1, Netscape Navigator v. 6.22 implementuje CSS Level 2 a DOM Level 2, apod.) se autoři rozhodli pro implementaci prostředí, jež bude provozovatelná ve WWW prohlížečích verzí MS Internet Explorer v. 6 a Netscape Navigator v. 6.22 a vyšších verzích (jež jsou dostupné na běžně užívaných platformách).
- *Web a aplikační vrstva* implementace bude realizována v prostředí aplikačního serveru IBM WebSphere s podporou technologie J2EE v. 1.3 (IBM WebSphere je certifikovaným aplikačním serverem pro J2EE v. 1.3 – viz. [8]). Využití technologie J2EE (zejména Java Servlets, Java Server Pages a Enterprise Java Beans viz. [2]) garantuje splnění požadavku portability softwarového řešení, s tím že bude organicky využito:
    - Technologie MVC při implementaci web, aplikační a databázové vrstvy
    - *Web vrstva* bude implementována pomocí Java Server Pages s výrazným podílem tzv. *custom tags* (příp. tag extension). Technologie custom tags umožňuje v rámci JSP stránek využít kromě standardních HTML tagů rovněž programátorem deklarované tzv. custom tags, jež jsou v rámci procesu generování dynamické WWW stránky převedeny pomocí jisté (často velice netriviální) transformace na množinu standardních HTML tagů. Použití technologie custom tags rovněž výrazným způsobem zvyšuje produktivitu tvůrce dynamických WWW stránek. V případě našeho projektu je tedy každé jednotlivé komponentě uživatelského rozhraní přiřazena množina (jednoho nebo více) custom tags, jež při generování dynamické WWW stránky zabezpečí její reprezentaci ve formě standardních HTML tagů.

Příklad. Jako krátký příklad využití technologie JSP custom tags může např. sloužit úvodní WWW stránka projektu informačního systému umožňující koncovému uživateli přihlášení do jejího prostředí – viz. obr. 2:



Obr. 2 – Přihlášení uživatele do prostředí informačního systému

Zdrojový tvar JSP stránky odpovídající WWW stránce na obr. 2 má pak tvar (jednotlivé custom tags jsou ve výpisu zvýrazněny, jejich význam je zřejmý):

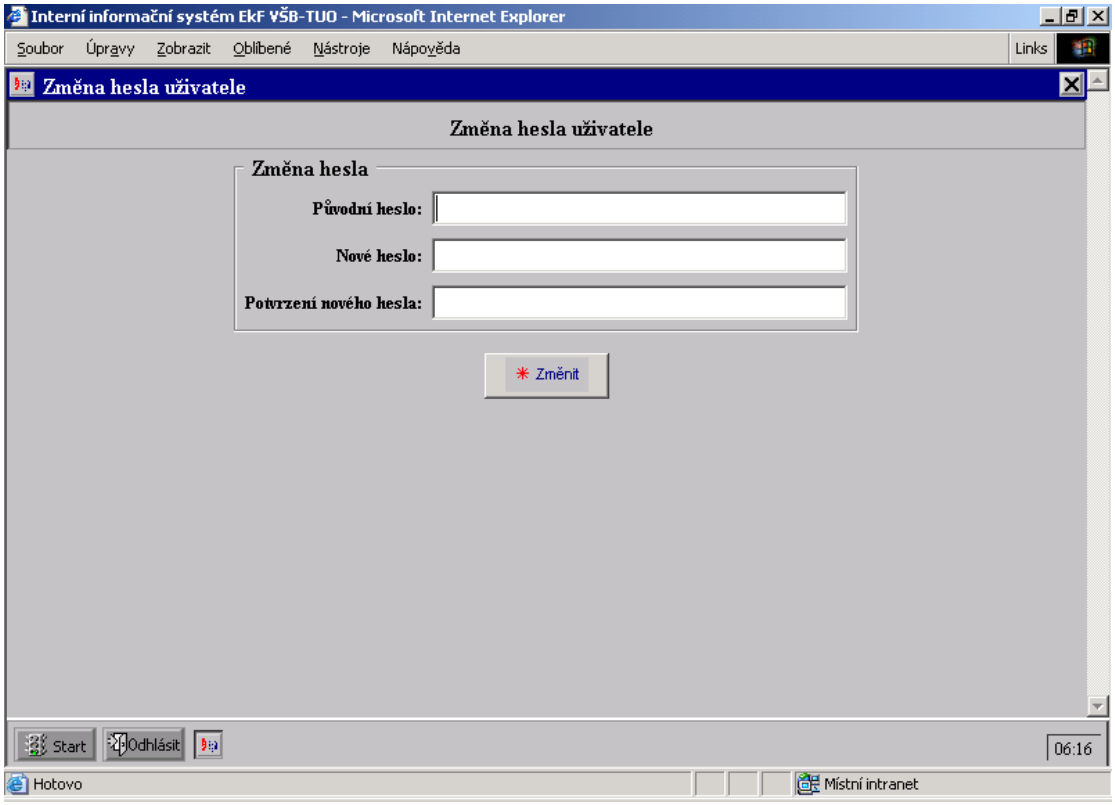
```
<%@page contentType="text/html; charset=windows-1250"%>
<%@taglib uri="isc.tld" prefix="vb" %>
<%@page import="cz.vsb.ekf.isc.*" %>

<vb:webPage headerText="Virtuální banka" background="#00BFFF" >
  <vb:form id="frm1" >
    <vb:basePanel background="#00BFFF" onMouseUp="o_inputBox.endMove();"
                  onMouseMove="o_inputBox.move( event.screenX, event.screenY
                  );" >
      <vb:dialogBox id="inputBox" width="450px" height="180" margin="4" left="150"
                    top="150" label="Virtuální banka" title="Virtuální banka"
                    cancelAction="window.close();" image="res/ekf.gif" >
        <vb:fieldset legend="Přihlášení uživatele" cellpadding="1" cellspacing="5" >
          <vb:inputText  id="logname" userName="Uživatelské jméno" width="300px"
                         label="Uživatelské jméno:" title="Uživatelské jméno"
                         labelPos="left" labelFontSize="10" minLength="1" />
          <vb:inputPassword id="passwd" userName="Heslo" width="300px" label="Heslo:"
                            title="Heslo" labelPos="left" labelFontSize="10"
                            minLength="5" />
        </vb:fieldset>
        <vb:buttonsRow top="10" >
          <vb:grButton id="loginBtn" type="<%= C.BTN_LOGIN %>" title="Přihlášení do
                       prostředí virtuální banky" action="makeLogin();" />
```

```
        <vb:grButton id="helpBtn" type="<%= C.BTN_HELP %>" title="Nápověda"
                                        action="showHelp()" />
      </vb:buttonsRow>
    </vb:dialogBox>
  </vb:basePanel>
</vb:form>

<Script Language="JavaScript">
  o_inputBox.show(); document.forms[1].e_logname.focus();

  function makeLogin()
  {
    if ( o_logname.doControls() && o_passwd.doControls() )
    {
      var f = document.forms[1];
      f.ev_logname.value = _convert( f.logname.value );
      f.ev_passwd.value = _convert( f.passwd.value );
      f.action = f.frmact.value + 'login?srv=1';
      f.target = '';
      f.submit();
    }
  }

  function showHelp()
  {
    var f = document.forms[1];
    f.action = f.frmact.value + 'login?srv=4';
    f.target = '_blank';
    f.submit();
  }
</script>
</vb:webPage>
```

Implementace kvalitní prezentační vrstvy na straně tenkého klienta je pak jedním z klíčových problémů celého projektu. Jak již bylo řečeno, stanoveným cílem je dosažení maximálně kvalitního uživatelského interface (včetně grafických komponent typu modal a modeless dialog box, pop-up menu, tree apod., jež nejsou standardně v rámci technologie HTML dostupné) na straně uživatele bez možnosti využití instalovaných plug-in komponent WWW prohlížeče (zejména technologie Java Applets, Flash apod.). Autory projektu byla tedy navržena a implementována poměrně rozsáhlá knihovna objektových komponent v programovacím jazyce JavaScript uživatelského rozhraní na straně tenkého klienta, jež rovněž respektuje technologii MVC, kde:

- *View* – je tvořen jednotlivými grafickými komponentami WWW stránek, které jsou definovány normami HTML, CSS a DOM
- *Controller* – je instancí příslušné třídy reprezentující *controller* grafické komponenty implementované v JavaScriptu
- *Model* – je instancí příslušné třídy reprezentující *model* příslušné grafické komponenty implementované v JavaScriptu

Uživatelské prostředí na straně tenkého klienta je pak navrženo v duchu designu známého z grafického interface standardních operačních systémů (Windows 2000, Linux), kde uživatelé pracují s jednotlivými moduly informačního systému podobným způsobem, jako s aplikacemi nad operačním systémem umístěnými v oknech grafického uživatelského rozhraní. (viz. obr. 3).



Obr. 3 – Prezentační vrstva informačního systému

Celé uživatelské prostředí informačního systému je tvořeno jedinou web stránkou, jednotlivá okna s aktivními moduly systému jsou realizována s využitím komponent FRAME (součást normy HTML), v rámci každého modulu je samozřejmě umožněna práce s několika vrstvami dialogových oken. Tento fakt má zásadní vliv na způsob implementace web a aplikační vrstvy informačního systému. Při reakci na událost strany tenkého klienta totiž není obecně možno dynamicky generovat novou web stránku a poté ji zaslat na stranu tenkého klienta – došlo by totiž k přepsání aktuální stránky na straně uživatele, která obecně může např. v okamžiku vzniku události obsahovat několik vrstev dialogových oken s rozpracovaným obsahem, příp. další elementy, jež by mohlo být při vygenerování nové stránky obecně velmi obtížné obnovit do stavu v okamžiku vzniku události. Ještě horším důsledkem, jež s sebou nese vygenerování nové web stránky, by obecně mohlo být přepsání aktuálního stavu prostředí informačního systému tvořeného jedinou web stránkou.

Z výše uvedených důvodů bylo nutno zcela změnit filozofii generování reakcí na události tenkého klienta, které realizuje web a aplikační vrstva systému. Díky důslednému použití technologie MVC jak na straně web a aplikační vrstvy, tak zejména na straně prezentační vrstvy tenkého klienta, ale nebylo nalezení řešení uvedeného problému vůbec obtížné. Stačí totiž, aby ***aplikační vrstva systému generovala jako reakci na událost v prezentační vrstvě tenkého klienta příslušný programový kód všech objektů typu controller příslušejících***

objektům typu view na straně tenkého klienta, jež se podílejí na reakci pro uvedenou událost. Tento způsob zabezpečuje:

- zachování aktuálního stavu uživatelského rozhraní tenkého klienta,
- možnost reagovat na událost současně několika objektům uživatelského rozhraní (pro jejich komponenty typu *controller* pak generujeme příslušný programový kód),
- realizovat virtuální prostředí umožňující současnou práci s několika moduly informačního systému (příp. externími informačními systémy),
- výrazné snížení objemu přenášených informací mezi stranou tenkého klienta a aplikační vrstvou systému – nejsou přenášeny obsahy celých web stránek, ale pouze reakce na událost strany klienta ve formě programového kódu objektů typu *controller*

Praktická realizace tohoto postupu s sebou tedy nese nutnost implementace aplikační vrstvy, jež jako výsledek reakce na událost strany klienta generuje programový kód v jazyce JavaScript, který bude na straně tenkého klienta (tedy na příslušné web stránce, jež není přepisována novým obsahem) interpretován příslušnými instancemi tříd typu *controller*. Úspora v objemu přenášených dat oproti klasickému způsobu generování dynamických web stránek může být podstatná a citelně uplatní se zejména u informačních systémů, jejichž klienti mají k dispozici pouze omezenou šířku pásma pro komunikaci s centrálním systémem.

4. Závěr

Závěrem příspěvku lze tedy konstatovat vysokou aktuálnost více než 20 let staré technologie Model-View-Controller při návrhu a implementaci informačních systémů s vícevrstvou architekturou podporující technologii tenkého klienta. Myšlenka přenosu kódu *controlleru* jako reakce na události ze strany tenkého klienta výrazně posunuje možnosti při návrhu a implementaci kvalitní prezentační vrstvy strany tenkého klienta.

Literatura:

1. BURBECK S.: Applications Programming in Smalltalk-80: How To Use Model-View-Controller, 1992, ParcPlace, http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html
2. http://java.sun.com/j2ee/
3. http://jakarta.apache.org/struts/index.html
4. http://jakarta.apache.org/
5. http://www.kontis.cz
6. http://www.w3.org/DOM/
7. http://www.w3.org/Style/CSS/
8. http://java.sun.com/j2ee/compatibility.html