TECHNOLOGIE MODEL-VIEW-CONTROLLER A JEJÍ UPLATNĚNÍ PŘI NÁVRHU A IMPLEMENTACI PORTÁLOVÝCH ŘEŠENÍ

Ivo Martiník

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

Abstrakt
Technologie tvorby portletů se v poslední době výrazně uplatňuje v oblasti realizace portálových řešení a přináší některé nové koncepty vzhledem k původní technologii servletů, která je široce využívána pro potřeby generování dynamických WWW stránek. Oficiální specifikace technologie portletů (JSR 168, Sun Microsystems) však neobsahuje některé významné koncepty, jež mohou mít při návrhu a realizaci portálových řešení zásadní význam. Jedná se např. o problematiku komunikace portletů, uplatnění technologie *Model-View-Controller* při jejich implementaci a další, jimž je věnován tento příspěvek.

1. Úvod

J2EE technologie (viz. [6]) se stále více uplatňují i v oblasti realizace komplexních portálových řešení. Portál lze v této souvislosti chápat jako soubor komponent a technologií, které umožňují zejména:

- Prezentaci statických i dynamických informací prostřednictvím WWW rozhraní s využitím technologie portletů (viz. [1])
- Personalizovaný přístup uživatelů k informacím
- Integraci aplikací
- Autentizaci a autorizaci včetně delegování administrace
- Propojení s externími informačními zdroji (databáze, adresářové služby)
- Řízenou publikaci informačního obsahu a jeho udržování v aktuální podobě

Portletem rozumíme web komponentu vytvořenou v programovacím jazyce Java, která není samostatně spustitelná a je po celou dobu svého životního cyklu umístěna v prostředí portletového kontejneru (portlet container), přijímá požadavky ze strany klienta a na jejich základě dynamicky generuje fragment WWW dokumentu. Fragmentem rozumíme část WWW dokumentu splňující jistá pravidla, jež spolu s ostatními fragmenty vytváří kompletní WWW dokument. Tento fragment vygenerovaný příslušným portletem je obvykle na úrovni portletového kontejneru (příp. portálu) sloučen podle přesně stanovených pravidel s fragmenty vygenerovanými dalšími portlety příslušnými danému WWW dokumentu. Portlety jsou v rámci portálového řešení tedy typicky využívány k vytváření prezentační vrstvy informačních systémů.

Koncepce portletů je v kontextu J2EE přitom velice úzce spojena s již několik let úspěšně využívanou technologií servletů (viz. [4]) a rovněž požadované vlastnosti portletového kontejneru se v mnoha bodech shodují s vlastnostmi softwarového kontejneru potřebného pro provoz servletů (a příp. Java Server Pages), portletový kontejner lze dokonce z tohoto pohledu chápat jako softwarovou vrstvu nad kontejnerem určeným pro běh servletů.

Mezi významnými rozdíly mezi technologiemi portletů a servletů lze zejména uvést:

- Výsledkem činnosti portletu v reakci na požadavek ze strany klienta není vygenerování úplného WWW dokumentu, ale pouze jeho fragmentu. Úplný WWW dokument je obecně výsledkem procesu agregace množiny těchto fragmentů na úrovni portálu do jednoho celku podle přesně stanovených pravidel. WWW dokument smí obecně obsahovat několik fragmentů vygenerovaných jedním portletem.
- Portlety nejsou přímo svázány s konkrétní URL adresou.
- Interakce WWW klienta s konkrétním portletem je prováděna nepřímo prostřednictvím portálu.
- S každým portletem je spojena množina jeho tzv. módů (View, Edit, Configure, příp. další) a stavů (Normal, Minimized a Maximized), které predikují jeho funkcionalitu a další charakteristiky v rámci WWW dokumentu.
- Portlety mohou uchovávat transientní data dvěma různými způsoby: na úrovni aplikace, jíž je portlet součástí, a na úrovni konkrétní instance portletu.

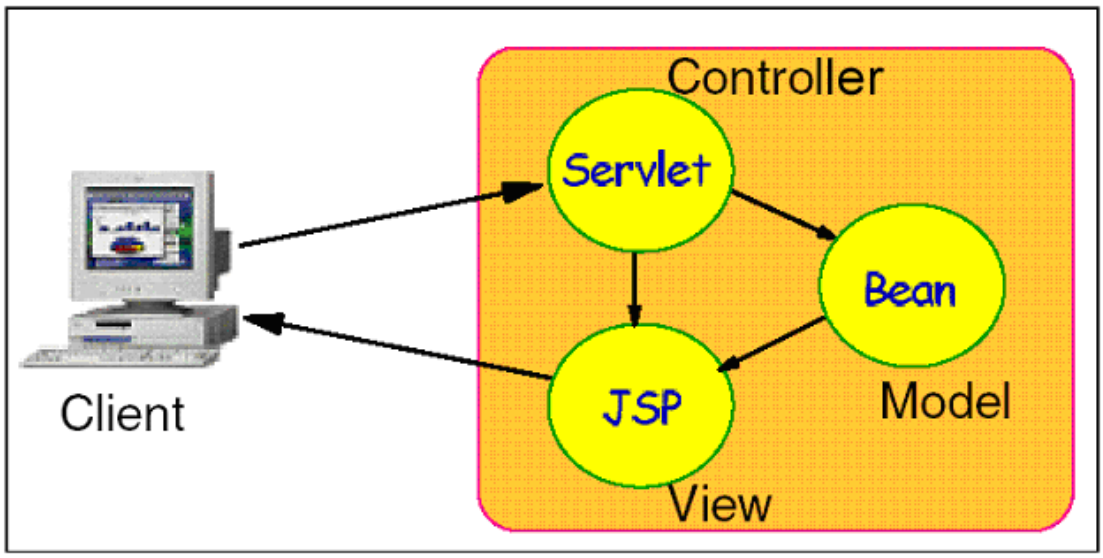
2. Technologie Model-View-Controller a její uplatnění při tvorbě portletů

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. [3]), 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) 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*.

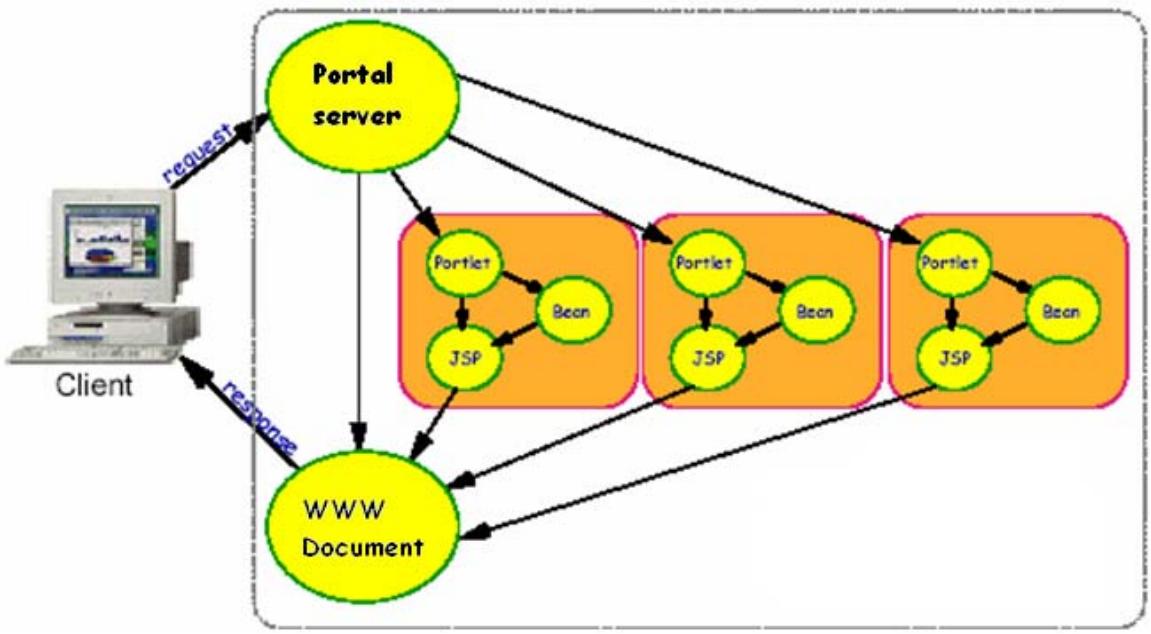
Technologii *Model-View-Controller* lze samozřejmě uplatnit i při návrhu a implementaci vícevrstvých distribuovaných programových systémů s využitím konceptů J2EE architektury, kde servlet tradičně vystupuje v roli *controlleru*, komponenta *view* je typicky implementována technologií Java Server Pages (JSP) a v úloze komponenty *model* nalezneme instanci třídy splňující požadavky normy pro Java Bean, příp. Enterprise Java Bean – viz. obr. 1.



Obr. 1: Model-View-Controller v prostředí J2EE architektury

Nasazení konceptu MVC při návrhu a implementaci portálových řešení s využitím technologie portletů má samozřejmě svá specifika. Koncept komponenty *model* přitom zůstává prakticky beze změny, pro potřeby aplikační logiky je obdobně jako v případě technologie servletů implementována instance třídy Java Bean, příp. Enterprise Java Bean. V roli komponenty *controller* však již vystupuje instance portletu, která provádí interakci s komponentami *model* a *view*. Výrazným posunem oproti technologii servletů je však skutečnost, že s každým portletem je spojen jeho aktuální mód (View, Edit, Configure apod) a stav (Normal, Minimized, Maximized). Každý portlet si tedy obecně v souvislosti se svým konkrétním stavem a módem smí zvolit konkrétní instance komponent *model* a *view*, s jejichž pomocí je generován fragment WWW dokumentu. Typické je zejména použití rozdílných instancí komponenty *view* pro generování konkrétního fragmentu příslušného aktuálními módu portletu.

Největší změny však při budování portálového řešení zcela jistě nalezneme v případě koncepce komponent *view*, která již ztratila kontrolu nad generováním úplného obsahu WWW dokumentu, ale jejím výsledkem je pouze jeho fragment. Obdobně jako v případě využití technologie servletů, jsou i v případě portletů standardně použity koncepty Java Server Pages (příp. HTML) fragmentů doplněné o technologii tzv. custom tags, které usnadňují proces generování fragmentů WWW stránky. Součástí API jsou rovněž pro tento účel implementované knihovny speciálních tagů. Při tvorbě komponent *view* je nutno samozřejmě dodržovat jistá pravidla pro generování fragmentů, zejména např. v případech vytváření úseků kódu ve skriptovacích jazycích (např. JavaScriptu) na úrovni jednotlivých fragmentů a případných kolizí a chyb (např. shodnost identifikátorů proměnných v různých fragmentech) ve výsledném dokumentu. Na úrovni portálu je poté zabezpečen proces agregace vygenerovaných fragmentů do jediného WWW dokumentu – viz. obr. 2.



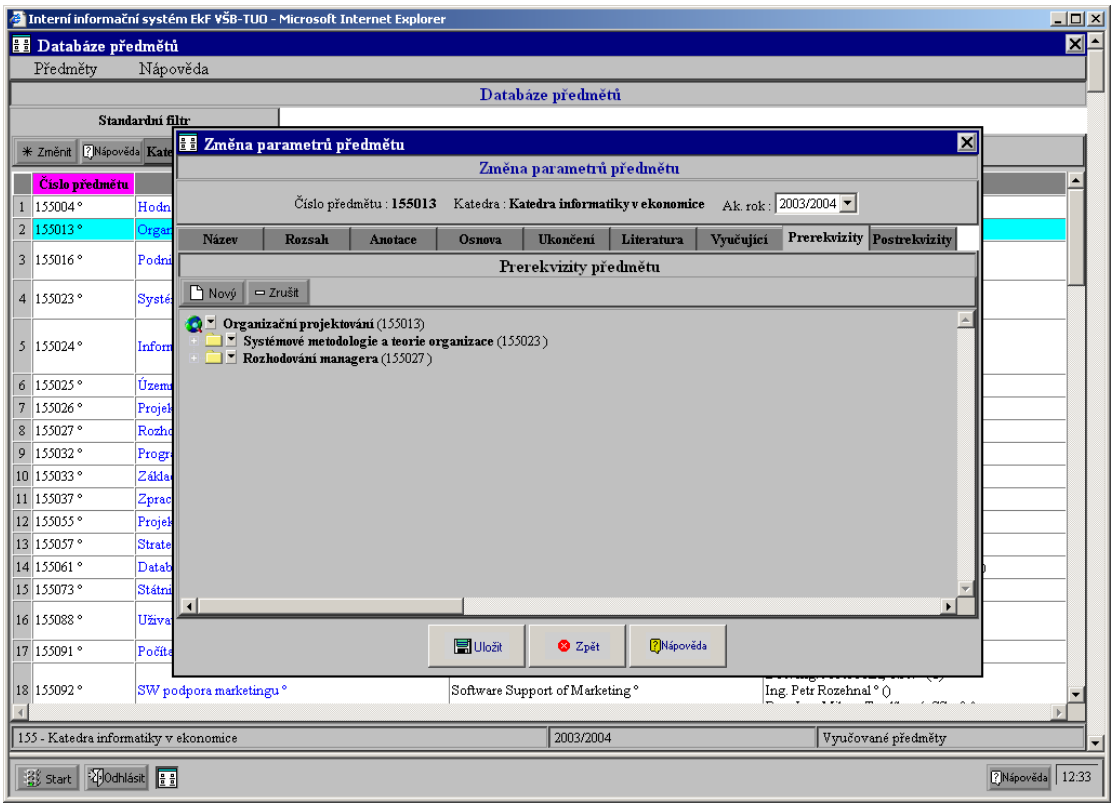
Obr. 2: Model-View-Controller v prostředí portálových řešení

3. Technologie Model-View-Controller a její uplatnění při implementaci fakultního portálu

V době vzniku příspěvku je na pracovišti jeho autora realizován projekt fakultního portálu, který bude organickou součástí celoškolského portálového řešení a jež v mnohém rozšiřuje koncepty původního návrhu informačního systému, jak byly prezentovány např. v [5]. Technická koncepce řešení předpokládá důsledné oddělení veřejné a neveřejné části portálu s možností aplikování individuálních a skupinových práv pro přístup k jednotlivým aplikacím.

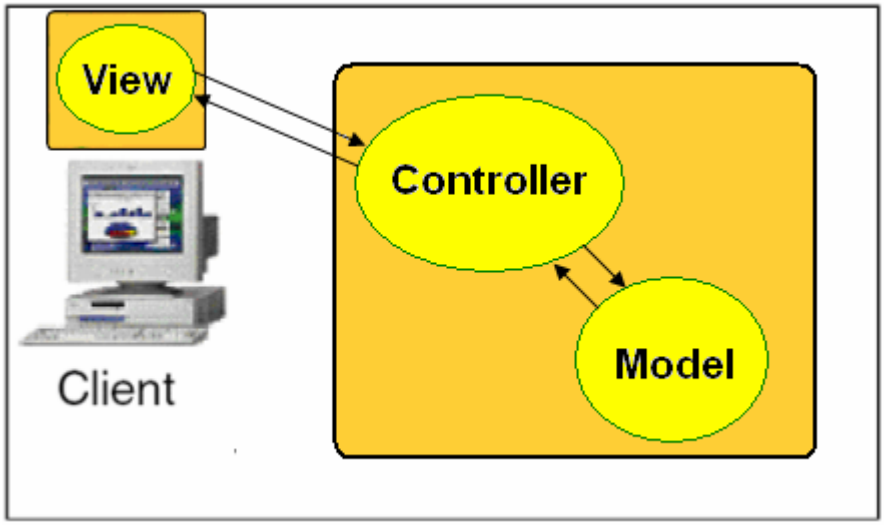
Pro implementaci neveřejné části portálu jsou využity následující technologie:

- *Prezentační vrstva* řešení je implementována s využitím technologií HTML, JavaScript, Document Object Model (DOM – viz. [7]) a Cascading Style Sheets (CSS – viz. [8]). 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. 7.1 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. 7.1 a vyšších verzích (jež jsou dostupné na běžně užívaných platformách) s cílem dosažení maximálního uživatelského komfortu připomínajícího práci v grafickém prostředí operačního systému podporujícího současný běh několika úloh – viz. obr. 3.
- *Web a aplikační vrstva* implementace je realizována v prostředí portálového serveru IBM WebSphere Portal Server (viz. [2]) s podporou technologie J2EE v. 1.3 a portletů.
- *Databázová vrstva* je tvořena databázovým systémem IBM DB2.



Obr. 3: Model-View-Controller v prostředí portálových řešení

Pro vlastní implementaci neveřejné části portálu je využita pro tento projekt speciálně vyvinutá distribuovaná varianta technologie *Model-View-Controller* (viz. obr. 4), kde v roli komponenty *view* vystupuje instance třídy reprezentující konkrétní element grafického rozhraní na straně WWW prohlížeče implementovaná v programovacím jazyce JavaScript. Ta prostřednictvím protokolu HTTP komunikuje s komponentou *controller*, jež je standardně realizována ve formě instance servletu na straně serveru. Vzájemná komunikace uvedených komponent je obousměrná a *controller* ve své podstatě reaguje na události strany tenkého klienta a ve spolupráci s komponentou *model* zpětně zasílá straně klienta výsledky těchto reakcí na události.

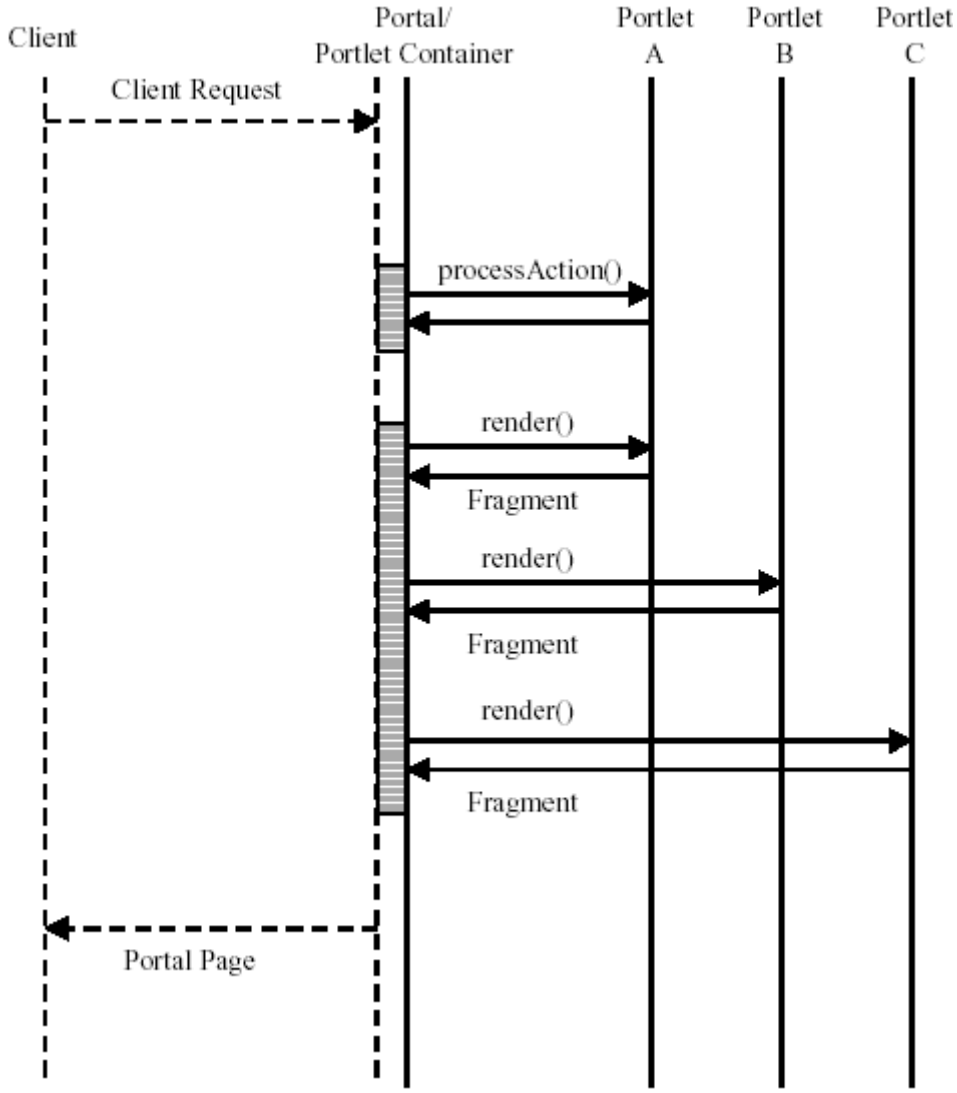


Obr. 4: Distribuovaná varianta technologie Model-View-Controller

Veřejná část portálu je implementována s využitím technologie portletů podle doporučení JSR 168 s aplikací principu *Model-View-Controller* způsobem popsaným v předchozích odstavcích.

Jedním z důležitých aspektů návrhu a implementace fakultního portálového řešení je problém kooperace portletů, který je v současné době ve specifikaci JSR 168 řešen jen velmi okrajově. Je zřejmé, že událost na straně tenkého klienta příslušná jistému fragmentu WWW dokumentu může vyvolat nutnost změny při generování ostatních fragmentů dokumentu. Portlet, jemuž přísluší reakce na danou událost, musí tedy jistým způsobem kooperovat s ostatními portlety podílejícími se na generování daného WWW dokumentu. Konkrétní situace řešení reakce na událost strany klienta může být ještě navíc determinována např. aktuálními stavy a módy uvedených portletů, které se mohou v této souvislosti měnit. Celá situace reakce na událost strany klienta je znázorněna na obr. 5 (viz. [1]). WWW dokument je zde agregován ze tří fragmentů, které jsou jednotlivě generovány instancemi portletů A, B a C, přičemž v první fázi reaguje portlet A na událost strany klienta voláním metody `processAction()`. Po ukončení volání této metody zabezpečí portletový kontejner vygenerování WWW dokumentu voláním metod `render()` jednotlivých participujících portletů A, B a C. Doporučení JSR 168 přitom nijak nestanoví pořadí volání metod `render()`, jejich běh může dokonce probíhat paralelně v rámci několika programových vláken.

Vlastní kooperaci portletů lze implementovat s využitím velice jednoduchých mechanismů, např. prostřednictvím sdílení hodnot jistých proměnných na úrovni aplikace (příp. ve společné databázi), které jsou k dispozici všem participujícím portletům. Komplikovanějším způsobem kooperace je programová realizace synchronních událostí spojených s předáním zprávy mezi participujícími portlety, které lze implementovat způsobem známým u technologie Java Beans v rámci jedné instance Java Virtual Machine. Je zřejmé, že celý mechanismus vyvolání událostí a reakcí na ně ze strany participujících portletů je nutno realizovat a ukončit ještě před zahájením fáze generování výsledného WWW dokumentu na úrovni kontejneru – tedy v rámci volání metody `processAction()` a před prováděním libovolné z metod `render()`- viz. obr. 5.



Obr. 5: Proces vytváření agregovaného WWW dokumentu

Další možností praktické implementace mechanismu kooperace portletů s využitím technologie MVC je sdílení společné komponenty *model* (příp. její sdílení ve vybraných módech participujících portletů), nebo obecněji zabezpečení kooperace několika komponent *model* příslušných různým portletům. Je-li komponenta *model* implementována prostřdnictvím technologie Enterprise Java Beans, lze jednoduše realizovat jejich synchronní i asynchronní komunikaci a dokonce tyto komponenty distribuovat mezi instancemi Java Virtual Machine.

4. Závěr

Závěrem příspěvku lze tedy konstatovat vysokou aktuálnost více než 20 let staré technologie *Model-View-Controller* při návrhu a implementaci portálových řešení s vícevrstvou architekturou. Zejména implementace distribuované varianty technologie *Model-View-Controller* výrazně přispívá ke zvýšení kvality prezentační vrstvy na straně tenkého klienta v prostředí fakultního portálu.

Literatura:

1. Abdelnur, A., Hepper, S.: Java Portlet Specification – Version 1.0 (JSR 168), Sun Microsystems, 2003
2. Rodriguez, J., Chan, S., Gonzales, B., Kroner, G., Parlangelo, M., Schwedler, S., Venancio, A.: IBM WebSphere Portal V5 – A Guide for Portlet Application Development – IBM Redbooks, 2003
3. 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
4. Hall, M.: More Servlets and Java Server Pages, Sun Microsystems Press, 2002, ISBN 0-13-067614-4
5. Martiník, I.: Technologie Model-View-Controller a její uplatnění při návrhu a implementaci informačních systémů v prostředí tenkého klienta
6. http://java.sun.com/j2ee/index.jsp
7. http://www.w3.org/DOM/
8. http://www.w3.org/Style/CSS/