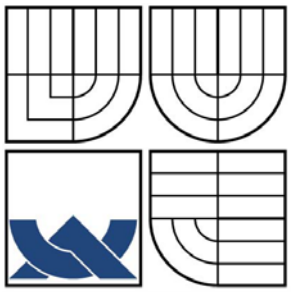
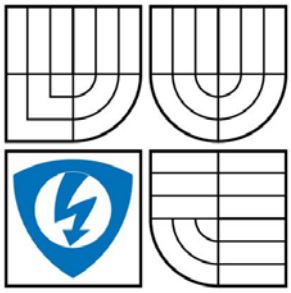


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

REALIZACE KOMPONENT V REDAKČNÍM SYSTÉMU JOOMLA

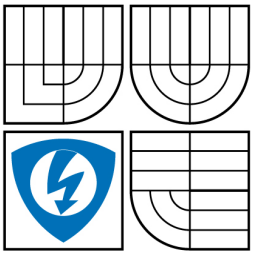
COMPONENT DESIGN FOR JOOMLA EDITORIAL SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MICHAL INGR
AUTHOR

VEDOUCÍ PRÁCE — ING. MARTIN KOUTNÝ
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Michal Ingr	***ID:***	78047
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Realizace komponent v redakčním systému Joomla

POKYNY PRO VYPRACOVÁNÍ:

Podrobně se seznamte s redakčním systémem Joomla. Prostudujte strukturu tohoto systému, princip modulů, komponent a pluginů. Na základě poznatků navrhněte a realizujte komponentu, která bude zastupovat elektronickou knihovnu s možností uživatelské komunikace. Realizujte jak veřejnou část systému, tak neveřejnou část systému. Při realizaci se snažte postupovat modulárně tak, aby bylo možné komponentu jednoduchým způsobem spravovat a rozšiřovat.

DOPORUČENÁ LITERATURA:

[1] KOSEK, Jiří. PHP - tvorba interaktivních internetových aplikací : podrobný průvodce. 1. vyd. Praha : Grada Publishing, 1998. 492 s. ISBN 80-7169-373-1.
[2] OPPEL, Andrew. Databáze bez předchozích znalostí : průvodce pro samouky. Přeložil David Krásenský. 1. vyd. Brno : Computer Press, 2006. 319 s. Obsahuje rejstřík. ISBN 80-251-1199-7.
[3]LEBLANC, Joseph L. Learning Joomla! 1.5 Extension Development : Creating Modules, Components, and Plugins with PHP. [s.l.] : Packt Publishing, 2007. 176. ISBN 978-1847191304.

Termín zadání:	9.2.2009	***Termín odevzdání:***	2.6.2009

Vedoucí práce: Ing. Martin Koutný

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práve třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Abstrakt

Bakalářská práce popisuje vytvoření a implementaci komponenty pro správu elektronické knihovny do redakčního systému Joomla. V úvodu jsou vysvětleny základní pojmy, jako například co je to komponenta nebo redakční systém. Zbytek práce se již zaměřuje na samotné vytvoření dané komponenty. Výsledná aplikace umožňuje uživateli provádět základní operace, jakými jsou například přidávání a mazání záznamů v systému. Celá aplikace je naprogramována v jazyce PHP s použitím některých vnitřních funkcí redakčního systému Joomla a propojena s databází MySQL. Cílem této práce je především vytvoření volně šiřitelného softwaru, který bude možné dále upravovat a rozšiřovat.

Klíčová slova

Joomla, systém pro správu obsahu, CMS, komponenta, PHP

Abstract

The bachelor's thesis describes the creation and implementation of component for the management of electronic library in the Joomla content management system. In the introduction, there are explain the basic concepts, such as what is a component or CMS. The rest of the work is focused on the creation of its own component. The final application allows user to execute basic operations such as adding and deleting records in the system. The entire application is programmed in PHP language with the use of certain internal Joomla content management system functions and interact with MySQL database. The main aim of this work is to create a free software, which can be further modify and extend.

Keywords

Joomla, content management system, CMS, component, PHP

INGR, M. *Realizace komponent v redakčním systému Joomla.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 45 s. Vedoucí bakalářské práce Ing. Martin Koutný.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma „Realizace komponent v redakčním systému Joomla“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne …… ……..

(podpis autora)

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Martinu Koutnému, zaměstnanci Ústavu telekomunikací, za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce.

V Brně dne …… ……..

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZRATEK

CMS – Content Management Systém
GNU – General Public Licence
PHP – Hypertext Preprocesor
IIS – Internet Information Services
HTML – HyperText Markup Language
CSS – Cascading Style Sheets
URL – Uniform Resource Locator
RSS – Really Simple Syndication

ÚVOD

Bakalářská práce se zabývá redakčním systémem Joomla, jeho strukturou a realizací komponenty. Joomla je bezplatný open source CMS určený pro publikování informací ve formě webových stránek na internetu nebo intranetu. Je napsaný v jazyce PHP a ke své činnosti využívá například databázi MySQL, JavaScript či Ajax. Podporuje například indexaci stránek, RSS, blogy, tisknutelné verze stránek, diskusní fóra, lokalizace či vícejazyčné verze.

Práce je rozdělena do dvou hlavních částí. První z nich se zabývá redakčním systémem Joomla, přibližuje jeho strukturu – rozdělení na uživatelskou a administrátorskou část (tzv. front end a back end). Následuje stručný popis jednotlivých částí. U front endu je to rozvržení a obsah jeho hlavní stránky, přehled nejčastěji používaných prvků. U back endu zase nejdůležitější položky v menu, jako plug-iny, komponenty, moduly, šablony či panely nástrojů.

Druhá část popisuje vytvoření samotné komponenty. Začíná založením vhodné adresářové struktury a registrací do systému Joomla. V několika podkapitolách následuje popis naprogramování funkcí pro přidávání, editaci a mazání záznamů v administrátorské části. V uživatelské části je využito již hotových funkcí, které ovšem musely projít drobnými úpravami a byla vytvořena nová funkce sloužící k zobrazování uložených dat.

1 JOOMLA!

Joomla je open source CMS (Content management system, česky systém pro správu obsahu nebo častěji také redakční systém) nabízený v mnoha jazycích včetně češtiny, který umožňuje komukoli vytvářet a upravovat obsah intranetových či internetových stránek bez nutnosti znát jakýkoliv programovací či značkovací jazyk. Je licencována pod GNU, takže je zdarma. Ke své činnosti potřebuje skriptovací jazyk PHP, databázový server MySQL a webový server Apache (možná je i instalace na server IIS). Samotné jádro systému mnoho funkcí nemá, lze je však rozšířit pomocí modulů, komponent či plug-inů, kterých existuje nespočet, a tak si každý může vymodelovat systém ke svému obrazu [1, 7].

1.1 Historie

Joomla se postupem času vyvinula z jiného redakčního systému a to z CMS australské firmy Miro pojmenovaným Mambo díky odtržení části vývojářského týmu kvůli vnitřním neshodám v roce 2006. Následuje stručný přehled:

Rok 2000 – společnost Miro vyvíjí CMS Mambo s nápadem rozdělení na uživatelskou a administrátorskou část.

Rok 2001 – Mambo ve verzi 3 se stává open source softwarem.

Rok 2002 – vychází komerční verze Mambo 2002 a Mambo 4 beta (ta už ovšem pod jiným vývojářským týmem) s přepracovaným schématem a architekturou kódu.

Rok 2003 – Mambo verze 4.5.

Rok 2004 – Mambo verze 4.5.1 s mnoha novými funkcemi a výrazným vylepšením administrátorské části.

Rok 2005 – Mambo verze 4.5.2 s lepší funkčností a stabilitou, objevují se spory uvnitř vývojářského týmu, jehož část odchází a přivádějí na svět Joomlu 1.0.

Rok 2007 – Joomla 1.5.

Oba systémy jsou si podobné a jejich vývoj pokračuje dál, Joomla se nachází v aktuální verzi 1.5.10, Mambo zase 4.6.5 [8].

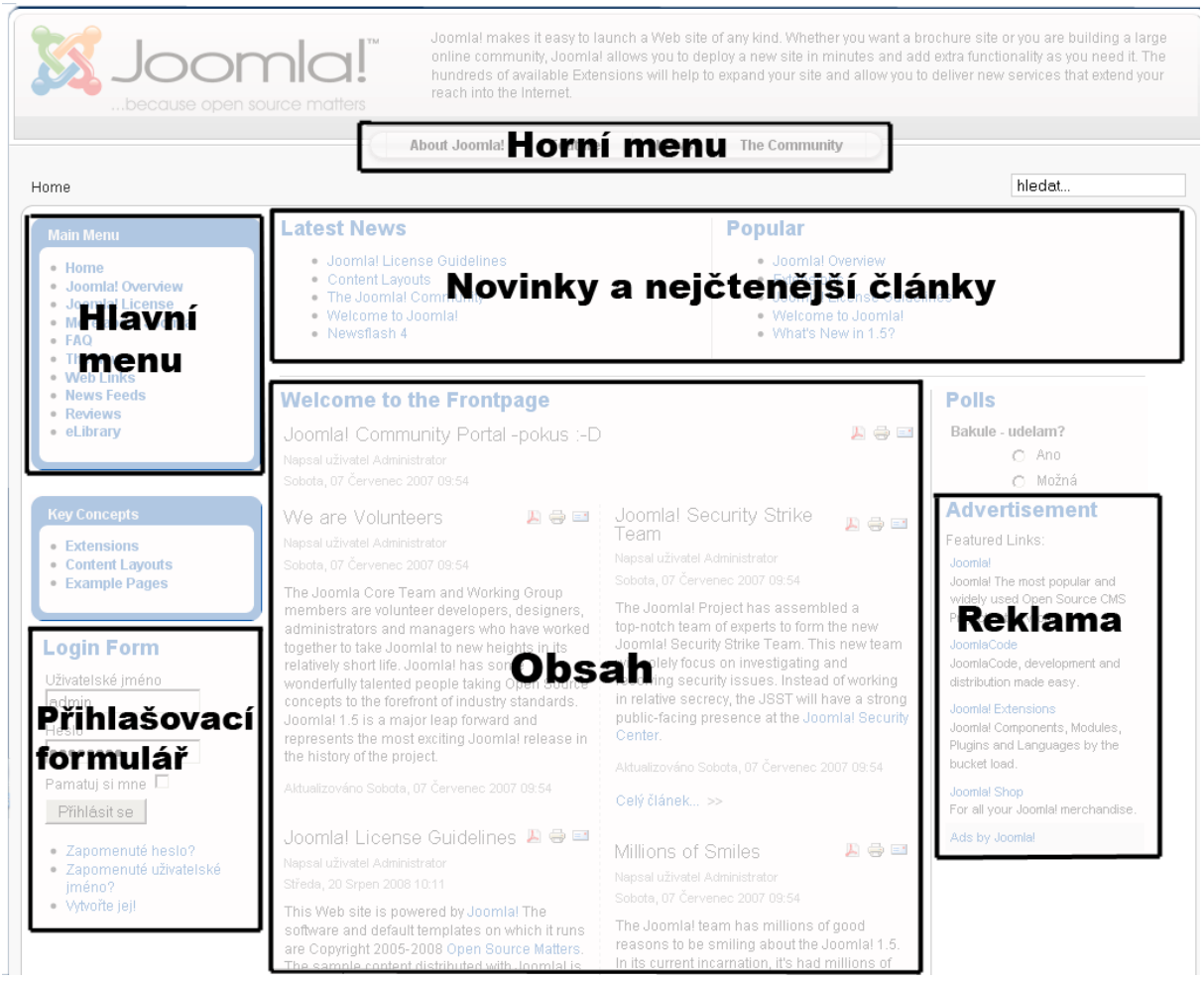
1.2 Redakční systém (CMS)

Redakční systém je software, který nějakým způsobem spravuje data. Má vstupní a výstupní rozhraní (back end a front end), kde vstup slouží pro vkládání dat ať už textového

nebo jiného charakteru a výstup zajišťuje jejich prezentaci. Vstupní část je podobná textovým editorům, kde můžeme psát články, vkládat obrázky nebo tvořit celé galerie, chaty, diskusní fóra či internetové obchody bez nutnosti znát jakýkoliv programovací jazyk. Možnosti jsou prakticky neomezené, vše záleží pouze na použitých komponentách nebo modulech. Naopak výstupní část už je klasická webová stránka, jejíž vzhled závisí na použité šabloně, do které je zapsaný text následně vložen a automaticky danému vzhledu přizpůsoben [2, 7].

1.3 Front end

Po nainstalování Joomly a otevření její domovské stránky je hned vidět, že nabízí mnoho funkcí a možností pro organizaci našeho webu. Stránka je rozdělena do několika sekcí (viz. obr.1), každou z nich lze samostatně editovat, přesouvat či jinak upravovat pomocí back endu.



Obr.1: Hlavní strana veřejné části.

1.3.1 Hlavní a horní menu

Hlavní menu je hlavní navigační oblastí na webu. Na každé stránce by se mělo vždy zobrazovat na stejném místě a vždy by tu měl být odkaz pro navrácení uživatele na hlavní stránku našeho webu. Horní menu se nachází na stránce tak vysoko, jak jen to nahraná šablona dovoluje. Je navrženo tak, aby uživateli poskytovalo rychlý přístup k nejdůležitějšímu obsahu na webu [7].

1.3.2 Novinky a nejčtenější články

Pro zobrazení těchto zpráv nabízí Joomla mnoho možností. Jak je vidět, jsou umístěny nad hlavním obsahem stránky. Je to z toho důvodu, že mezi uživateli často patří k nejvyhledávanějším. Joomla sama kontroluje, který článek či zpráva byla přidána naposledy a která je nejčtenější a podle toho automaticky aktualizuje tuto sekci [7].

1.3.3 Reklama

Pokud se naše stránka stane populární, můžeme prodávat reklamní prostor nebo si jen s někým vyměnit bannery, k čemuž slouží právě toto místo. Reklamní banner bývá nejčastěji ve formátech jpg, png nebo pohyblivý gif či swf a ve formě odkazu. Reklama nemusí být umístěna na stejném místě jako na obrázku 1, ale třeba na úplném konci stránky či pod přihlašovacím formulářem[7].

1.3.4 Přihlašovací formulář

Jak už název napovídá, tento modul slouží pro přihlašování uživatelů. Ti jsou rozděleni do několika skupin s různými právy a možnostmi úprav webu. V případě zapomenutí nebo ztráty hesla či přihlašovacího jména si je můžeme nechat zaslat na e-mail. Lze využít také k registraci nového uživatele [7].

1.3.5 Obsah

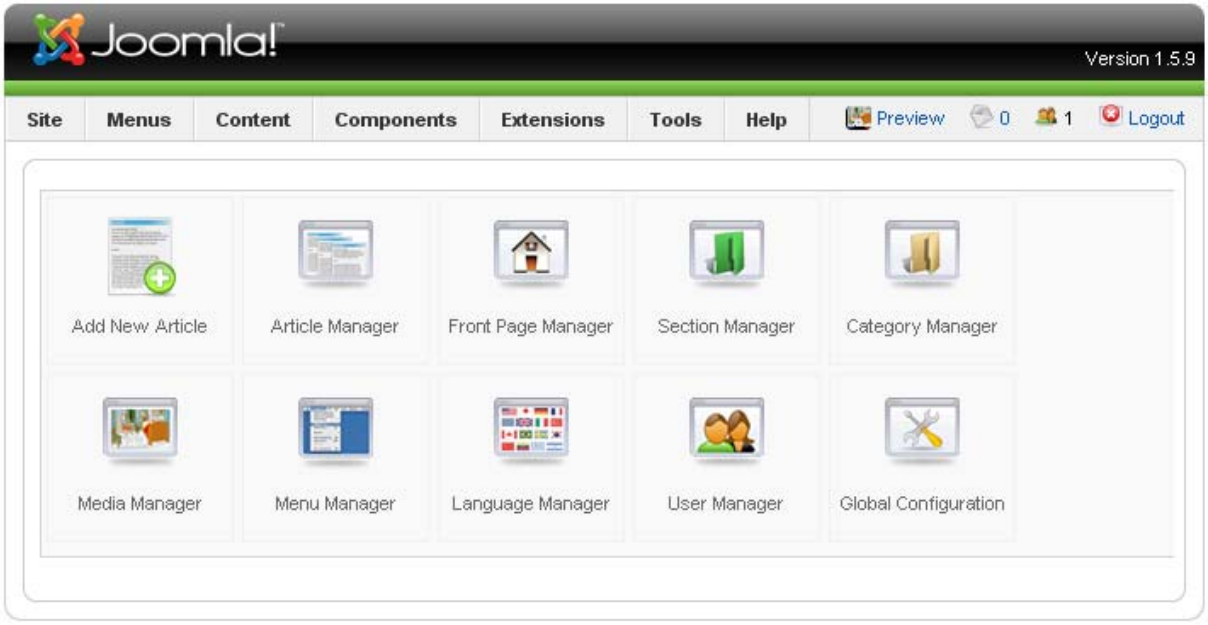
Zde se zobrazuje nejdůležitější část webové stránky – její obsah. Joomla nabízí mnoho způsobů zobrazení, záleží jen na uživateli jaké schéma si zvolí. Můžeme nechat zobrazovat jen titulky nebo části textu, nechat si článek vytisknout nebo převést do pdf atd. [7].

1.3.6 Ostatní

Jak je z obrázku 1 vidět, Joomla nabízí mnohem víc než jen těch pár zmíněných funkcí. Můžeme přidávat ankety, různá počítadla přístupů, doplňovat či vytvářet další menu a vůbec vše, co nás napadne.

1.4 Back end

V této části systému se provádí veškerá administrace, je rovněž rozdělena do několika sekcí a menu podle toho, kterou část stránky budeme zrovna upravovat (viz. obr.2). Přístup sem mají jen uživatelé s právy Manager a vyšší.



Obr.2: Hlavní strana administrátorské části.

1.4.1 Menu

Stejně jako v uživatelské části je i zde menu nejdůležitějším prvkem stránky. Pomocí něj se dostaneme ke všem nástrojům pro správu webu. Je rozděleno na několik části (viz. obr.2) a vpravo od něj se nachází další čtyři položky. Preview představuje odkaz na uživatelskou část webu, takže si po změně obsahu můžeme ihned prohlédnout výsledek. Druhá ikona upozorňuje na příchozí zprávy ve vaší schránce. Třetí ukazuje aktuální počet přihlášených uživatelů a čtvrtá slouží pro odhlášení z administrace.

1.4.2 Panel nástrojů

Pokud vybereme nějakou položku z menu zobrazí se nám u ní její panel nástrojů (viz. obr.3). Ten obsahuje různé ikony v závislosti na tom, jaké operace s ní lze provádět. Nejpoužívanější jsou ikony pro vytvoření nové položky, její uložení, editaci a mazání.

Obr.3: Panel nástrojů.

1.4.3 Komponenty

Komponenty jsou jednou z nejdůležitějších součástí Joomly. Jsou to aplikace, které rozšiřují její funkce, jako například správce reklamních bannerů, správce kontaktů, tvorba anket, odkazy, informační kanály a jiné, vše naprogramované v PHP. Joomla je navržena tak, aby se načetla a spustila pouze jedna komponenta pro danou generovanou stránku [3, 6].

1.4.4 Moduly

Moduly jsou úzce spjaty s komponenty. Na rozdíl od nich se může na stránce objevit libovolný počet modulů. Moduly tvoří například menu nebo formuláře, čili se starají o prezentaci dat. Takže když pomocí komponenty vytvoříme například anketu, tak modul nám zajistí kde a jakým způsobem se zobrazí [4, 6].

1.4.5 Plug-iny

Plug-in je kus programového kódu, který se připojuje do Joomly za účelem změny její funkčnosti. Může být použit například uvnitř textu pro načtení modulu nebo při vyhledávání či formátování výstupu (zvýraznění klíčových slov, komentářů apod.) [6].

1.4.6 Šablony

Šablona tvoří vizuální podobu stránek. Definuje barvy, znak písma, velikostí písma, pozadí, obrázky, řádkování a dělení na stránce. Jinými slovy všechno, co má co dočinění se vzhledem stránky. Šablona se skládá z nejméně jednoho HTML souboru pro strukturu stránky a jednoho CSS soubor pro design. Každý si tak může upravit vzhled podle svého nebo vytvořit vlastní [5].

1.5 Skupiny uživatelů a uživatelská práva

V Joomle jsou dvě hierarchické skupiny. Jedna řeší práva v uživatelské části systému, druhá v administrátorské. Pro front end jsou to tedy skupiny Registered, Author, Editor a Publisher. Pro back end Manager, Administrator a Super Administrator. Uživatelé ve skupině Registered se mohou na stránkách přihlásit a získat tak informace, které jsou přístupné jen registrovaným uživatelům. Nemají možnost jakkoliv měnit obsah webu. Uživatelé skupiny Author už mohou navíc přidávat své vlastní články a také je editovat. Před jejich zveřejněním však musí být schváleny někým s vyššími přístupovými právy. Uživatelé ve skupině Editor mohou také přidávat a editovat své vlastní články a navíc editovat články ostatních uživatelů, stále je však nemohou publikovat bez kontroly. Skupina Publisher má stejná práva jako Editor, ovšem navíc může zveřejňovat články předchozích skupin. Jako Manager máte stejná práva jako Editor, ale můžete také upravovat menu a obsah v administraci. Ve skupině Administrator máte navíc možnost instalovat a spravovat komponenty, moduly, prug-iny atd. Skupina Super Administrator má veškerá práva [7].

2 VYTVOŘENÍ KOMPONENTY

Vytvořené komponenty reprezentují jednoduchou elektronickou knihovnu obsahující čtyři části – monografii, elektronické články, příspěvky ve sborníku a webové stránky, které jsou provázány v jednotný celek. Popis se bude týkat pouze elektronických článků, u ostatních částí se postupuje analogicky.

2.1 Adresářová struktura

Joomla používá specifické schéma, které je společné pro všechny komponenty. Proto je nutné před jejich samotným programováním vytvořit složky, do kterých se budou skripty ukládat. Každá komponenta musí mít jedinečný název bez mezer. Zdrojové kódy jsou rozděleny do dvou částí, jedna je určena pro veřejnou část, tzv. front end, která se nachází v kořenovém adresáři ve složce `components`, druhá pro administrátorskou část, tzv. back end, která rovněž leží v kořenovém adresáři, ovšem ve složce `administrator/components` . Každá ze složek, do kterých komponenty ukládáme, musí začínat předponou `com_`, aby systém rozpoznal, že se jedná skutečně o komponentu a podle toho s ní zacházel. V tomto případě tedy ponese název `com_clanky`. Pokud je komponenta načtena z veřejné části, systém hledá soubor s názvem komponenty končící příponou `.php`. Proto ve složce `components/com_clanky` vytvoříme soubor `clanky.php`. Podobně pro administrátorskou část musí soubory začínat předponou `admin.`, čili v adresáři `administrator/components` založíme nový soubor `admin.clanky.php`.

2.2 Načtení komponenty

Všechny uživatelské dotazy se v Joomle provádí přes soubor `index.php` v kořenovém adresáři. Různé komponenty se načítají pomocí proměnné `option` zapsané v URL. Pokud si tedy nyní do prohlížeče zadáme adresu `index.php?option=com_clanky`, zobrazí se nám prázdná stránka. Abychom vyzkoušeli zda-li vše opravdu funguje, vložíme do souboru `clanky.php` následující kód:

```
<?php
defined( '_JEXEC' ) or die( 'Restricted access' );
echo '<div class="componentheading">Electronic article</div>'; ?>
```

Výsledek by měl vypadat jako na obrázku 4.

Obr.4: Úvodní obrazovka

Na začátku zavoláme funkci `defined()`, která nám zkontroluje, zda-li je komponenta skutečně volána Joomlou. Pokud vše proběhne v pořádku, Joomla sama nakonfiguruje prostředí s některými bezpečnostními opatřeními a vypíše text na třetím řádku [6]. Stejný kód můžeme vložit do souboru `admin.clanky.php` pro ověření funkce v administrátorské části, adresa bude ovšem trochu odlišná – `administrator/index.php?option=com_clanky`.

2.3 Registrace komponenty v databázi

Nyní víme, jak přistupovat ke komponentám ve front endu, tak i v back endu. Ovšem neustálé vypisování adresy je poněkud nepohodlné, proto registrujeme naši komponentu přidáním řádku do tabulky komponent. Joomla využívá databázi MySQL a všechny tabulky v ní uložené začínají předponou `jos_`. Pro registraci tedy použijeme následující dotaz:

```
INSERT INTO jos_components (name, link, admin_menu_link,
admin_menu_alt, option, admin_menu_img, params) VALUES (Electronic
Articles, option=com_clanky, option=com_clanky, Manage Articles,
com_clanky, js/ThemeOffice/component.png, '');
```

Můžeme také samozřejmě využít grafické prostředí phpMyAdmin, pokud ho máme nainstalované. Zadané údaje budou stejné, neuvedené nulové nebo prázdné. Tím dáme Joomle základní informace o naší komponentě – její jméno, adresy pro veřejnou i administrátorskou část, název složky, kde budou uloženy zdrojové soubory, a grafickou podobu v menu back endové části, takže nemusíme sami nic programovat, systém už se o to postará sám. Pokud vše proběhlo v pořádku, měli bychom dostat výsledek jako na obrázku 5.

Obr.5: Registrovaná komponenta v databázi

Pokud je komponenta skutečně zaregistrována, můžeme na ni vytvořit odkaz v menu veřejné části. Přihlásíme se tedy do back endu, kde vybereme Menus -> Main Menu, klikneme na New a v rozbaleném seznamu pak na Electronic articles. Zde stačí vyplnit jméno, pod kterým se položka objeví v menu. Můžeme si ale pohrát i s ostatním nastavením, jako například možnosti zobrazení položky pouze registrovaným uživatelům, otevírání odkazu v novém okně apod.

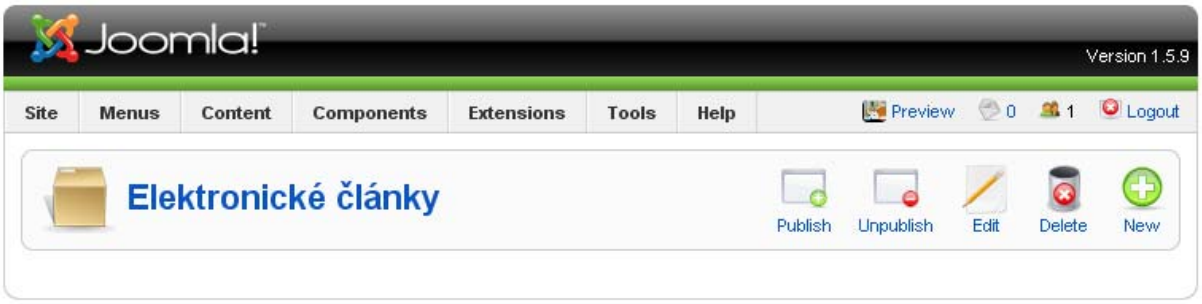
2.4 Vytvoření panelu nástrojů

V administrátorské části mají standardní komponenty předdefinován panel nástrojů s tlačítky pro ukládání, mazání, editaci a publikování položek. Joomla nám umožňuje použít tyto tlačítka i pro vlastní komponenty, čehož využijeme a ušetříme si tak nějaký kousek práce navíc. Vytvoření panelu nástrojů je rozděleno do dvou souborů a to z důvodu oddělení vizuální a funkční části od sebe, což v budoucnu zjednoduší případné úpravy. Aby vše fungovalo jak má, vytvoříme tedy nejdříve ve složce `administrator/components/com_clanky` soubor `toolbar.clanky.html.php`. Soubory obsahující výstupní kód jsou obvykle uspořádány do tříd. V tomto případě také vytvoříme třídu pojmenovanou `TOOLBAR_clanky`, která obsahuje několik funkcí, kde každá z nich představuje různé tlačítko. Třída `JToolBarHelper` naopak obsahuje všechny funkce, které vygenerují

nezbytný HTML kód pro zobrazení těchto tlačítek [6]. Samozřejmě pokud bychom se rozhodli, můžeme naprogramovat vlastní výstup, ale je výhodnější použít již předdefinované funkce, kde máme zaručenu jejich kompatibilitu s redakčním systémem. Nyní, když máme nadefinovaný panel nástrojů, potřebujeme přidat nějakou podmínku, která rozhodne, které z tlačítek se mají zobrazit. V back endu Joomla automaticky nahraje soubor začínající názvem komponenty a končící `clanky.php`. Takže ve stejné složce vytvoříme další soubor nesoucí název `toolbar.clanky.php`, ve kterém se nachází následující kód:

```
<?php
defined( '_JEXEC' ) or die( 'Restricted access' );
require_once( JApplicationHelper::getPath( 'toolbar_html' ) );
switch($task)
{
  case 'edit':
  case 'add':
  TOOLBAR_clanky::_NEW();
   break;
  default:
  TOOLBAR_clanky::_DEFAULT();
   break;
} ?>
```

Řádek obsahující `require_once()` používá funkci `getPath()` z třídy `JApplicationHelper`, což nám umožňuje načtení `toolbar.clanky.html.php` nezávisle na názvu komponenty, takže pokud se někdy v budoucnu rozhodneme komponentu přejmenovat, funkce nám stále načte správný panel nástrojů [6]. Pokud vše dopadlo dobře, měli bychom vidět výsledek stejný jako na obrázku 6.



Obr.6: Základní panely nástrojů

2.5 Naprogramování administrátorské části

Nyní, když máme komponentu registrovanou v databázi a systém o ní ví a máme také hotový panel nástrojů, přiřadíme jednotlivým tlačítkům jejich funkce a dokončíme back endovou část naší aplikace.

2.5.1 Vytvoření tabulky v databázi

Předtím, než vytvoříme rozhraní pro vkládání článků, je třeba založit tabulku, do které budou vlastní záznamy ukládány. Použijeme tedy buď grafické rozhraní prostředí PhPMyAdmin nebo SQL dotaz. Tabulka bude obsahovat jedinečný identifikátor každého řádku (tzv. primární klíč), údaje o autorovi článku, jeho název, jazyk a název periodika, rok vydání, údaje o datu citování, prosto pro komentář od zadavatele, ukazatel do tabulky pro uložení souboru, údaje o zadavateli, pole pro rozhodnutí, zda-li bude položka veřejná či soukromá a jestli bude vůbec publikována (viz tab.1). Také hned vytvoříme druhou tabulku `jos_knihovna_soubory`, která bude sloužit k uložení záznamů o přiloženém souboru a z tabulky `jos_clanky` se do ní budeme odkazovat. Bude obsahovat pouze dva sloupce, jedním bude opět primární klíč a druhý bude obsahovat název souboru.

jos_clanky	
id	int(11)
name	varchar(255)
surname	varchar(255)
title	varchar(255)
language	varchar(255)
periodical_title	varchar(255)
date_of_issue	varchar(20)
day	tinyint(4)
month	tinyint(4)
year	tinyint(4)
comment	text
uploaded_file	mediumint(9)
user	varchar(255)
public	tinyint(1)
published	tinyint(1)

Tab.1: Tabulka pro ukládání článků.

Nyní, když máme hotové tabulky pro ukládání dat, je zapotřebí obstarat jednotlivé funkce pro provádění dotazů na databázi. K tomu použijeme již hotové funkce třídy `JTable`, kterou pro nás připravili programátoři Joomly. Tato třída poskytuje funkce pro vytváření,

čtení, aktualizace a mazání záznamů z jedné tabulky v databázi. Proto budeme muset vytvořit jednu jak pro tabulku `jos_clanky`, tak pro `jos_knihovna_soubory`. Přesuneme se tedy do adresáře `administrator/components/com_clanky`, kde vytvoříme složku `tables`, do které vložíme soubory `clanek.php` a `soubor.php`. Posledně jmenovaný bude obsahovat následující kód:

```
<?php
defined('_JEXEC') or die('Restricted access');
class TableSoubor extends JTable
{
 var $id = null;
 var $soubor_id = null;
 function __construct(&$db)
   {
   parent::__construct( '#__knihovna_soubory', 'id', $db );
   }
} ?>
```

Funkci `defined()` není třeba vysvětlovat, setkali jsme ses ní již v úvodu. Na dalším řádku vytváříme rozšíření třídy `JTable` o naši vlastní, čímž zajistíme, že budeme moct používat všechny její předdefinované funkce. Dále nastavíme všechny sloupce z tabulky jako členy proměnných a přiřadíme jim nulovou hodnotu. Nakonec pomocí funkce `__construct()` vytvoříme konstruktor (=speciální metoda třídy, která se volá, když je instance příslušného objektu této třídy nově vytvářena) pro třídu, která přijímá odkaz na aktuální instanci databáze. Ta to funkce vyžaduje rodičovský konstruktor, který potřebuje název tabulky, sloupec s primárním klíčem a databázovou instanci. Tím zajistíme, že třída `TableSoubor` zdědí funkce jako `store()`, `load()`, `delete()` a další, které umožňují spravovat záznamy v databázi, takže nemusíme psát žádné SQL dotazy [6].

2.5.2 Vytvoření formuláře pro ukládání dat

Nyní, když máme hotovou tabulku pro ukládání dat, musíme je do ní nějakým způsobem dostat a jak jinak, než přes formuláře. Stejně jako při vytváření panelu nástrojů, tak i zde musíme oddělit výstupní HTML kód (soubor `admin.clanky.html.php`) od zpracovatelských funkcí (soubor `admin.clanky.php`). Začneme tedy druhým jmenovaným, do kterého vložíme následující část kódu:

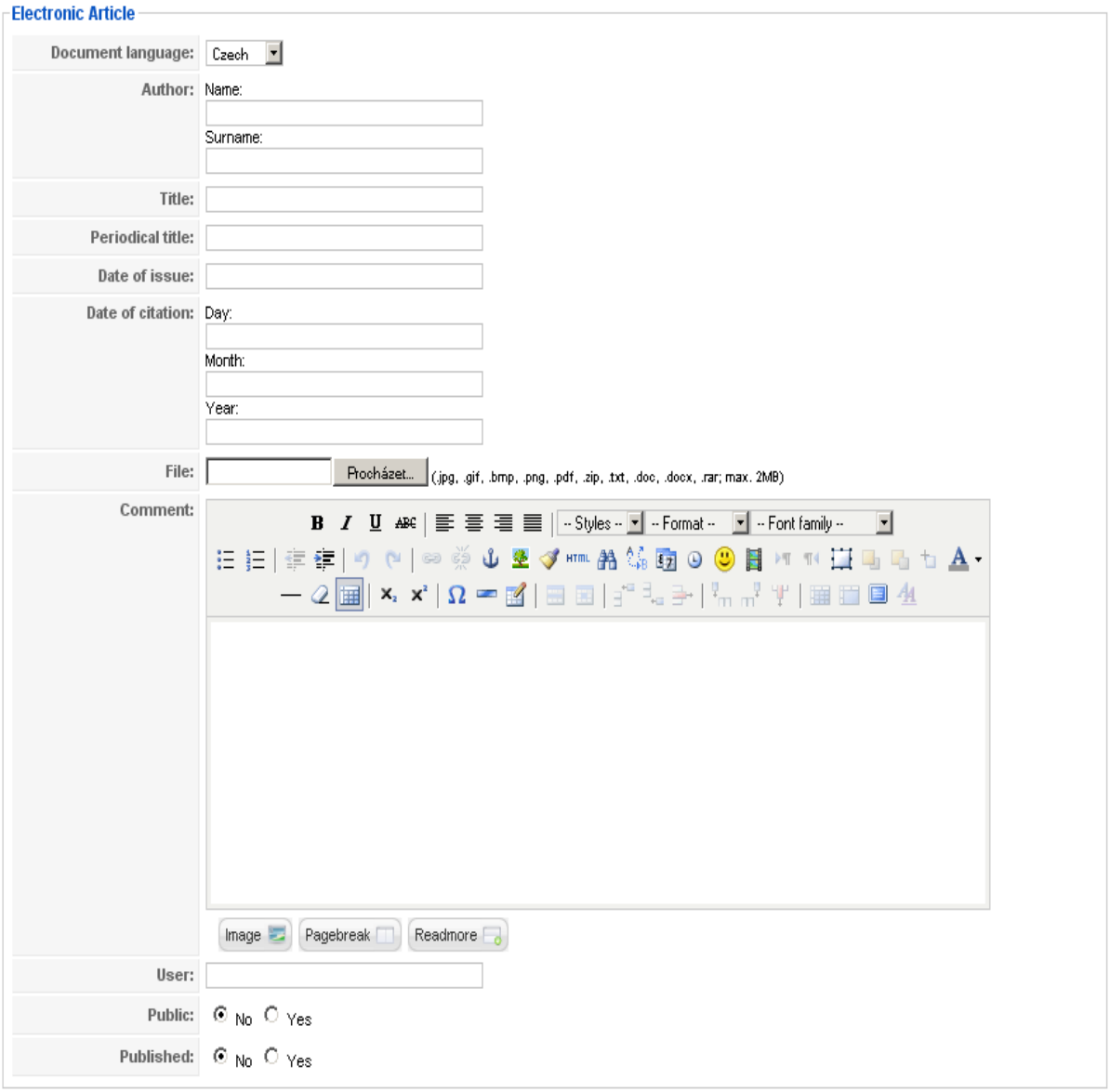
```
<?php
defined( '_JEXEC' ) or die( 'Restricted access' );
require_once( JApplicationHelper::getPath( 'admin_html' ) );
JTable::addIncludePath(JPATH_COMPONENT.DS.'tables');
switch($task)
{
 case 'add':
 editClanky( $option );
 break;
}
function editClanky( $option )
{
$row =& JTable::getInstance('clanek', 'Table');
$lists = array();
...
} ?>
```

Na prvním řádku máme již dobře známou funkci `defined()` pro ověření volání skriptu Joomlou, hned po ní následuje `require_once()` pro vložení souboru `admin.clanky.html.php`. Funkce `getPath()` načítá řetězce jako jsou `admin_html`, `front_html` a třídy a vrací absolutní cestu k odpovídajícím souborům. Přesto, že jsme v kódu neuvedli název složky, `getPath()` stále zajistí vložení správného souboru i kdybychom změnili název komponenty nebo HTML souboru [6]. Dále potřebujeme obstarat obsluhu databáze za použití námi vytvořené třídy z předchozí kapitoly. To nám zajistí funkce `addIncludePath()` z třídy `JTable`. Tato funkce automaticky vloží soubory nadefinované v adresáři `tables [9]`.

Poté přidáme rozhodovací blok. Použit byl příkaz `switch()`, který nejprve vyhodnotí testovaný výraz v podobě proměnné `$task` a poté prochází jednotlivé větve příkazu a ignoruje veškeré příkazy, dokud nenarazí na shodu hodnoty testovaného výrazu a hodnoty dané větve. Jakmile se tedy v proměnné bude nacházet hodnota „add“ spustí se funkce `editClanky()`. Ta nejprve načte příslušnou tabulku z databáze pomocí funkce `getInstance()`, následuje `getVar()` z třídy `JRequest` pro načtení hodnoty zadaných proměnných získaných metodou `GET`, `POST` nebo `REQUEST` [6]. Pro nahrání příslušného řádku z tabulky slouží funkce `load()` z třídy `JTable`. Do ní vkládáme hodnotu sloupce, který jsme zadali v rodičovské funkci `__construct()` souboru `clanek.php` v adresáři

`tables`. Následuje příprava některých HTML prvků předtím, než je zobrazíme pomocí funkce `HTML_clanky::editClanky()`.

Nyní přidáme do souboru `admin.clanky.html.php` kód pro zobrazení formulářových polí nutných pro ukládání záznamů do tabulky. Za zmínku také stojí funkce `JHTML::_()`. Tato funkce automaticky generuje některé formulářové prvky, jako jsou například check box, radio button nebo list box, takže je nemusíme vytvářet sami [6]. Výsledek je vidět na obrázku 7.



Obr.7: Formulář pro ukládání článků.

Podíváme-li se do zdrojového kódu vidíme, že začínáme vytvořením nové třídy s názvem `HTML_clanky`, ta bude sloužit i pro některé další funkce vždy, když budeme chtít zobrazit jejich výstup. Následuje samotná funkce `editClanky()`, které pomocí parametrů předáváme, mimo jiné, některé předdefinované formulářové prvky z předchozího odstavce. Další řádek obsahuje funkci `JFactory::getEditor()` vloženou do proměnné `$editor`, která vrací odkaz na globální editor objektů. Poté již následuje samotný formulář formátovaný do tabulky, který musí být pojmenován „adminForm“, jinak nám nebude fungovat dříve vytvořený panel nástrojů. Za zmínku ještě stojí řádek `<?php echo $editor->display( 'comment', $row->comment , '100%', '250', '40', '10' ); ?>`, kde funkce `display()` zobrazí výše zmíněný textový editor. Pokud ovšem není nadefinován, zobrazí se místo něj pouze obyčejné textové pole [6]. Obsahuje následující prvky: název proměnné, do které budou data uložena, její hodnotu, výšku a šířku zobrazeného editoru, počet sloupců a řádků. Poslední dvě hodnoty slouží určení rozměrů textového pole, pokud by se nepoužil textový editor.

2.5.3 Ukládání dat

Formulář pro zadávání dat máme hotový, teď musíme ošetřit funkci tlačítka Save v našem panelu nástrojů, kterým vložená data uložíme do databáze, takže v souboru `admin.clanky.php` naprogramuje další funkci – `saveClanky()`.

Napřed však vytvoříme funkci pro ukládání souborů, která proběhne ještě před samotným uložením článku do databáze. Začneme nejprve ověřením, zda-li vůbec nějaký soubor nahráváme. Pokud ano, definujeme proměnnou typu pole, do které uložíme přípony souborů, které bude možné na server nahrát. Dalším krokem je zjištění přípony právě nahrávaného souboru. K tomu využijeme několika PHP funkcí pro práci s řetězci. První z nich je `substr()`, která vrací část řetězce. Můžeme zadat tři parametry (dva jsou povinné a jeden nepovinný). Prvním parametrem je řetězec, který má být analyzován, v našem případě tedy jméno nahrávaného souboru. Druhým parametrem je pozice na které bude začínat vrácený řetězec, přičemž pozice prvního znaku je 0. My jako druhý parametr použijeme funkci `strpos()`, která vrací pozici hledaného textu v řetězci. Jejím prvním parametrem je prohledávaný řetězec, tedy opět název nahrávaného souboru, druhý text, který v něm hledáme, čili tečku před příponou souboru. Nyní se dostáváme k třetímu parametru funkce `substr()`, kterým je celé číslo, jež udává délku podřetězce, který má být vrácen.

K jeho zjištění je použita další funkce pro práci s řetězci a to `strlen()`, která vrací délku zadaného řetězce, v našem případě opět jména nahrávaného souboru. Shrnuto, načteme název souboru, pomocí funkce `strpos()` zajistíme načítání znaků od tečky před příponou souboru včetně a nakonec načteme samotnou příponu [10].

Když tedy máme načtenou příponu souboru přidáme podmínku, ve která zjišťujeme, zda-li je daný typ souboru povolen:

```
if(!in_array($ext,$allowed_filetypes)){
        die('The file you attempted to upload is not allowed.');
       }
```

K tomu použijeme funkci `in_array()`, která má dva parametry. Prvním je hledaná hodnota, tedy proměnná s příponou nahrávaného souboru, druhým prohledávané pole, čili pole, ve kterém máme uloženy vybrané přípony. Pokud je hledaná hodnota nalezena, funkce vrací true a skript pokračuje dál, pokud ne, vrátí false vykoná se příkaz `die()`, který vypíše chybovou hlášku a přeruší vykonávání skriptu [11]. Následuje podmínka pro omezení velikosti nahrávaného souboru a to z toho důvodu, aby se nám ze serveru nestalo nějaké úložiště dat. Pokud je soubor menší než maximální povolená velikost, proběhne jeho zpracování. Nejprve se připojíme na tabulku s uloženými soubory. Použijeme již předdefinovanou funkci `JFactory::getDBO()`, která vrací právě probíhající spojení s databází [6]. Do proměnné `$query` vložíme dotaz na databázi:

```
$query = "SELECT * FROM #__knihovna_soubory";
```

Příkazy v něm uváděné již nejsou z jazyka PHP, ale SQL. Příkaz `SELECT` slouží pro prohledávání databáze, hned za ním musí následovat seznam položek, které chceme z tabulky přečíst. Znak „*“ znamená, že chceme vypsat veškerý její obsah. Direktiva `FROM` určuje odkud mají být data načtena, tedy z jaké tabulky. Poté je volána funkce `setQuery()`, které je jako parametr předán dotaz na databázi, který je zde uchován pro pozdější použití. Následuje volání funkce `loadObjectList()`, ta spustí provádění onoho dotazu na databázi a jednotlivé řádky tabulky jsou načteny do proměnné `$rows` typu pole jako objekty [6]. K jejich zpracování použijeme PHP funkci `foreach()`, která je určena k procházení polí:

```
foreach($rows as $row)
  {
    while ($i<$row->id):
    $i++;
    endwhile;
  }
```

Funkce prochází polem `$rows`, v každém cyklu je hodnota aktuálního prvku přiřazena proměnné `$row` a interní ukazatel pole je zvýšen o jedničku (takže v dalším cyklu budeme pracovat s dalším prvkem). Když se `foreach()` spouští poprvé, je interní ukazatel pole automaticky nastaven na první prvek pole, není tedy potřeba napřed volat funkci `reset()` [12]. Pracuje také pouze s kopií specifikovaného pole, nikoliv s polem samotným, a proto se změny v původním poli neprojeví. Pomocí cyklu `while()` zjistíme nejvyšší hodnotu primárního klíče. To děláme z toho důvodu, abychom zjistili, na jaký název budeme muset nahrávaný soubor přejmenovat. Tím, že každý nově nahraný soubor přejmenujeme, zabráníme možnému opakování se názvů souborů a s tím spojených problémů. Máme tedy zjištěnu nejvyšší hodnotu prvku v tabulce a můžeme soubor přejmenovat. Přičteme tedy k této hodnotě jedničku a přidáme příponu souboru, kterou máme stále uloženou v proměnné pro testování povoleného typu souboru. Teď musíme nový název uložit do tabulky `jos_knihovna_soubory`. Nejprve nastavíme proměnnou `$row` jako instanci naší třídy `TableSoubor` pomocí funkce `getInstance()`, do proměnné pro název souboru vložíme jeho nové jméno a zavoláme funkci `store()`, která nám vše uloží do databáze. Nakonec ještě zbývá někam nahrát onen soubor, který máme zatím uložen pouze v paměti. Pro určení cesty použijeme předdefinovanou funkci Joomly `JPATH_BASE`, která vrátí absolutní cestu ke složce, kam byla Joomla nainstalována, k ní přidáme adresář, do kterého chceme soubor nahrát a nakonec ještě samotné jméno souboru:

```
$newname = JPATH_BASE.'/tmp/'.$filename;
move_uploaded_file($_FILES['uploaded_file']['tmp_name'],$newname);
```

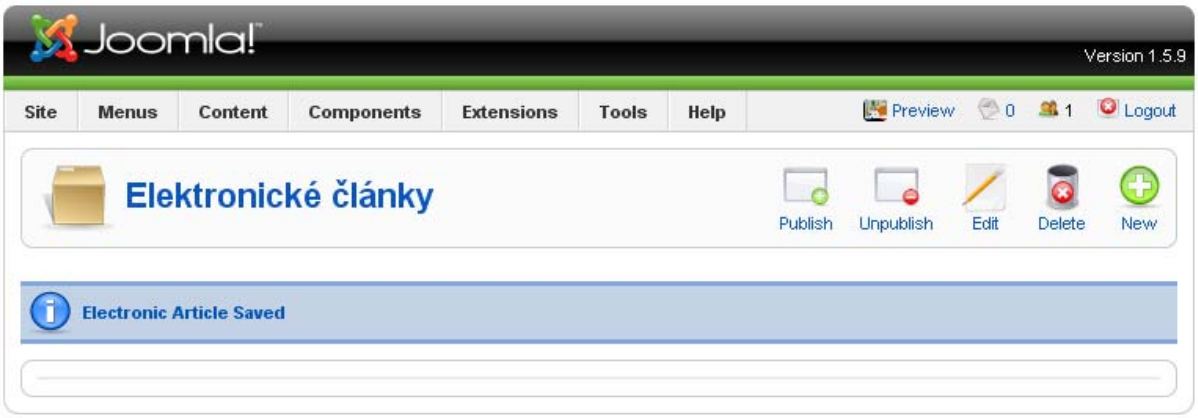
Funkce `move_uploaded_file()` pak zajistí samotný přesun souboru na určené místo. Ta má dva parametry, prvním je jméno nahrávaného souboru v dočasném adresáři a cestu, do které bude soubor uložen. Někdo by možná chtěl použít funkci `copy()`, která se pro práci se soubory používá. To však není možné, protože po jeho nahrání metodou `POST` byl uložen do dočasného systémového adresáře a z důvodu restrikce přístupu je s ním možné manipulovat pouze pomocí funkce `move_uploaded_file()`. Pokud vše proběhlo jak mělo, měl by být soubor na serveru i uložen v databázi a můžeme se pustit do programování funkce pro ukládání článků – `saveClanky()`.

V ní nejprve nadefinujeme globální proměnnou `$mainframe` a právě navázané spojení s databází. Proměnná `$mainframe` obsahuje mnoho funkcí, které slouží pro práci s objektem `Session` (slouží pro ukládání dat) a h`eader` (hlavičky). Následuje podmínka pro zjištění, zda-li nahráváme nějaký soubor. Pokud ano, proběhne podobná akce, jako

při nahrávání souboru. Připojíme se na databázi a zjistíme nejvyšší hodnotu primárního klíče v tabulce `jos_knihovna_soubory`, kterou si uložíme do proměnné `$filename`. Poté nastavíme proměnnou `$row` jako instanci naší třídy `TableClanek` pro zajištění uložení dat do správné tabulky. Funkce `bind()` přiřadí hodnoty, resp. vlastnosti, pole do tabulky objektu [6]. Abychom ochránili databázi SQL před tzv. injection attack, čili útokem na databázi, prováděným přidáním některých příkazů do dotazu na databázi, voláme funkci `JRequest::get()`. Tato funkce zkontroluje všechny hodnoty proměnných předávaných metodou `POST` z formuláře pro ukládání článku a vypustí z nich znaky, které by mohly být použity pro daný útok. Pokud se funkce `bind()` z nějakého důvodu nepodaří provést, zobrazí se chybová hláška pomocí JavaScriptu a vrátí uživatele na předcházející stránku. Po přiřazení hodnot už můžeme manipulovat přímo s proměnnými uložených v objektu `$row`.

Vzhledem k tomu, že proměnná `comment` může obsahovat HTML tagy, které by mohly být automaticky odstraněny z důvodu bezpečnosti aplikace, musíme s ní zacházet jinak než s ostatními. K tomu použijeme funkci `JRequest::getvar()`, do které zadáme název proměnné, výchozí hodnotu, pole, z kterého chceme vzít její hodnotu, předpokládaný datový typ proměnné a příznak `JREQUEST_ALLOWRAW`. Ještě přidáme hodnotu primárního klíče právě nahraného souboru do proměnné `uploaded_filename`, abychom se pak na něj mohli odkazovat a zavoláme funkci `store()`. Ta převezme všechny proměnné a přidá je do dotazu na databázi. V závislosti na hodnotě `id` se provede buď příkaz `UPDATE` (používaný pro editaci hodnot jednotlivých položek tabulky) nebo `INSERT` (určený pro vložení nového záznamu do tabulky). Protože však přidáváme nový záznam, není v proměnné `id` uložena žádná hodnota a provede se příkaz `INSERT` [6]. Pokud SQL dotaz vrátí chybu, vypíšeme ji na obrazovku a umožníme uživateli vrátit se na předcházející stránku, pokud ne, zavoláme funkci `redirect()`, která nás přesměruje na hlavní stránku naší komponenty a vypíše hlášku o úspěšném uložení hodnot do databáze (viz. obr.8). Ještě předtím však musíme do rozhodovacího bloku `switch()` přidat následující řádky, které zajistí spuštění příslušných funkcí:

```
case 'save':
 uploader($dir_upload);
 saveClanky( $option, $task, $uploaded_file );
 break;
```



Obr.8: Uložení zadaných dat.

Vložené záznamy se nám samozřejmě nezobrazí, protože není hotová funkce pro jejich načítání. Jediným způsobem je nechat si vypsat uložená data pomocí PhPMyAdmina (webové rozhraní pro správu databáze).

2.5.4 Vypsání záznamů z databáze

Protože neustálé přepínání se mezi naší aplikací a PhpMyAdminem kvůli prohlížení vložených záznamů není vůbec komfortní, naprogramujeme další funkci, která zobrazí uložená data přímo v komponentě. Otevřeme tedy soubor `admin.clanky.php` a přidáme následující kód:

```
function showClanky( $option )
  {
  $db =& JFactory::getDBO();
  $query = "SELECT * FROM #__clanky";
  $db->setQuery( $query );
  $rows = $db->loadObjectList();
  if ($db->getErrorNum()) {
    echo $db->stderr();
    return false;
    }
  HTML_clanky::showClanky( $option, $rows );
  }
```

Tato funkce nejdříve získá odkaz na právě probíhající spojení s databází. Do proměnné `$query` potom vložíme dotaz, který chceme na databázi vykonat a zavoláme funkci `setQuery()`. Ta převezme dotaz a uchová ho pro pozdější volání na databázi, nevykoná se ihned. Po spuštění funkce `loadObjectList()` je spuštěn onen dotaz

a jednotlivé řádky tabulky jsou uloženy do pole jako objekty. Pokud funkce vrátí nějakou chybu, je zastavena a chyba se vypíše na obrazovku. O to se stará funkce `getErrorNum()`, která vrací číslo chyby posledního dotazu a funkce `stderr()`, která chybu vypíše. Jestliže vše proběhlo v pořádku, předáme výsledky dotazu k zobrazení v souboru `admin.clanky.html.php`.

Zde funkce začíná vytvořením formuláře se jménem adminForm, je to z toho důvodu, abychom i tu mohli používat námi vytvořený panel nástrojů, jinak by tlačítka nereagovala. Následuje záhlaví tabulky, ve které budeme záznamy zobrazovat. Nejsou v ní uvedeny všechny položky uložené v databázi, ale pouze pár nejdůležitějších k získání informací o jaký záznam se jedná. Prvním z nich je ovšem check box, který je naprogramován tak, aby po zaškrtnutí označil veškeré zobrazené položky. Toho lze využít například při vymazání všech položek, nemusíme je označovat jednotlivě. Následuje cyklus `For`:

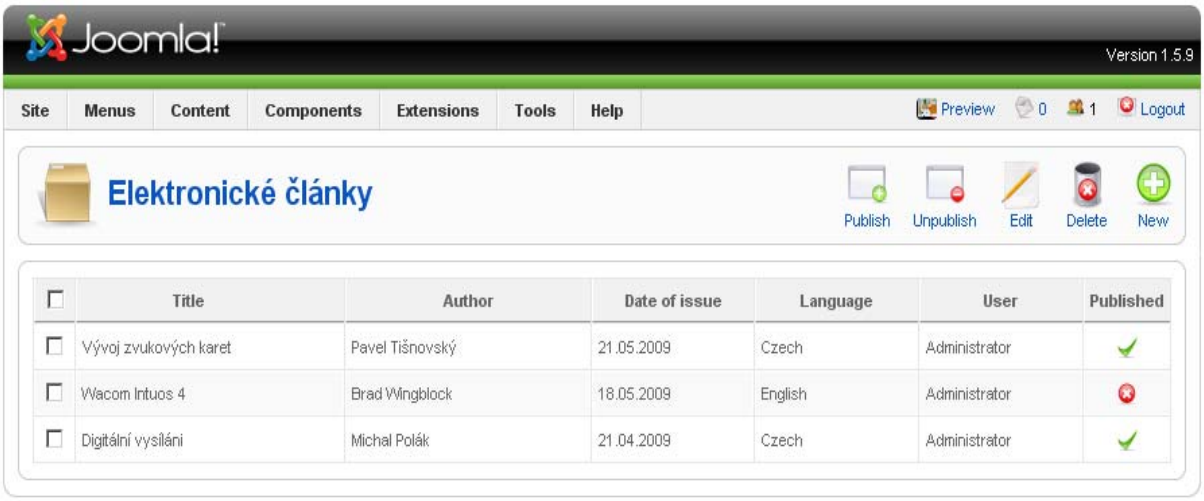
```
<?php
$k = 0;
for ($i=0, $n=count( $rows ); $i < $n; $i++)
{
  $row = &$rows[$i];
  $checked = JHTML::_('grid.id', $i, $row->id );
  $published = JHTML::_('grid.published', $row, $i );
?>
```

Ten vytvoří smyčku, která prochází jednotlivé řádky načtené z tabulky. Proměnná `$i` je nastavena na nulu, v proměnné `$n` je počet záznamů tabulky, takže cyklus běží tak dlouho, dokud máme stále co vypisovat na obrazovku. Uvnitř smyčky ukládáme do proměnné `$row` aktuálně načtený řádek, abychom mohli zobrazit jeho obsah. Některé hodnoty z daného řádku tabulky jsou zobrazeny přímo, ale jiné potřebují zvláštní zacházení (primární klíč a proměnná pro publikování článku). Použitím funkce `JHTML::_('grid.id')` zajistíme zobrazení primárního klíče jako check boxu a jeho propojení s JavaScriptem pro použití ve spolupráci s panelem nástrojů. Čili pokud označíme nějaký záznam a klikneme na tlačítko Delete, funkce zajistí předání správného `id` a daný řádek vymaže z databáze. Funkce `JHTML::_('grid.published')` zase generuje obrázek jako tlačítko závislé na hodnotě ve sloupci `published` naší tabulky. Pokud je jeho hodnota 1, vygeneruje obrázek zelené fajfky (publikováno), je-li 0, zobrazí se červený křížek (nepublikováno). Nyní už jen zobrazíme načtené záznamy do připravené tabulky. Pod ní se ještě nachází několik skrytých prvků. První z nich obsahuje hodnotu proměnné `option`, která uchovává název komponenty a slouží například pro přesměrování stránky po uložení nebo vymazání článku na stránku

se správnou komponentou. Proměnná `task` zase díky JavaScriptu umožňuje používat panel nástrojů bez nutnosti odeslání formuláře. Nakonec ještě zbývá přidat sekci `default` do rozhodovacího bloku `switch()` v souboru `admin.clanky.php`:

```
default:
  showClanky( $option );
  break;
```

To nám zajistí vypsání všech záznamů v tabulce při načtení komponenty (tzn. když je proměnná `task` prázdná). Nyní když přidáme několik záznamů, tak se nám zobrazí pod panelem nástrojů (viz. obr.9).



Obr.9: Zobrazení záznamů z tabulky.

2.5.5 Úprava záznamů

Prozatím jedinou možností jak změnit uložené údaje je přes správce databáze PhPMyAdmin jejich editováním přímo v tabulce, nebo, pokud nemáme přístup do databáze, jejich smazáním a opětovným zadáním. Ani jedna z možností neposkytuje patřičný komfort a tak si naprogramujeme vlastní funkci, respektive pouze upravíme již hotovou `editClanky()`.

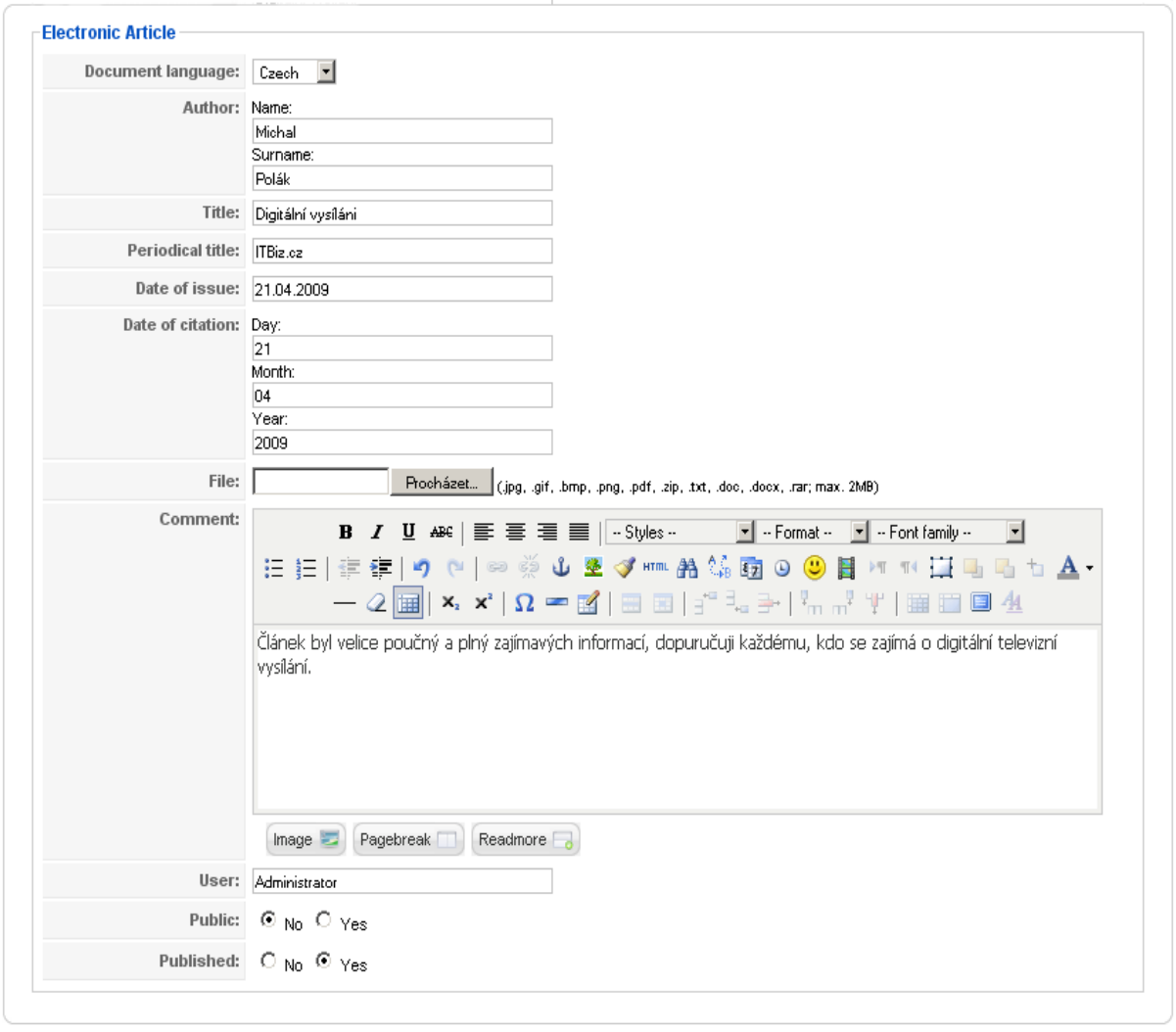
Nejprve přidáme pod řádek `$row =& JTable::getInstance('clanek', 'Table')` následující kód:

```
$cid = JRequest::getVar( 'cid', array(0), '', 'array' );
$id = $cid[0];
$row->load($id);
```

Přidali jsme proměnnou `$cid`, která je typu pole a obsahuje všechny primární klíče z tabulky. Protože však můžeme současně editovat pouze jeden záznam, nastavíme ji na první prvek pole, tj. na vybraný záznam a načteme ho z databáze pomocí funkce `load()`. V souboru `admin.clanky.php` ještě zbývá doplnit příkaz `case`:

```
switch($task)
{
  case 'edit':
  case 'add':
  editClanky( $option );
  break;
```

Když teď vybereme některý ze zobrazených záznamů a klikneme na tlačítko `Edit`, zobrazí se nám formulář pro zadávání dat s vyplněnými poli (viz. obr.10).



Obr.10: Editace záznamů.

Možná jste si všimli, že při editaci nebo přidávání nového záznamu máme v panelu nástrojů ještě tlačítko `Apply`. To umožňuje uložit právě prováděné změny v záznamu a dál v editaci pokračovat. K jeho zprovoznění musíme udělat několik změn v souboru `admin.clanky.php`. Jednu hned na začátku souboru v příkazu `switch()`, kde před klauzuli `case 'save'` přidáme další, `case 'apply'`. Druhou ve funkci `saveClanky()`, tam poslední řádek nahradíme následujícím kódem:

```
switch ($task)
    {
    case 'apply':
    $msg = 'Changes to Electronic Article saved';
    $link = 'index.php?option=' . $option .
    '&task=edit&cid[]='. $row->id;
    break;
    case 'save':
    default:
    $msg = 'Electronic Article Saved';
    $link = 'index.php?option=' . $option;
    break;
    }
  $mainframe->redirect($link, $msg);
```

Zde rozhodneme, jaký dotaz na databázi se vykoná. Buď tedy uložíme provedené změny tlačítkem `Apply` a pokračujeme dál v editaci, nebo se po uložení vrátíme zpět na hlavní stranu.

2.5.6 Mazání záznamů

Naprogramování této funkce je relativně jednoduché. Nejprve upravíme soubor `admin.clanky.php`, kde do příkazu `switch()` přidáme další větev:

```
case 'remove':
  removeClanky( $option, $uploaded_file );
  break;
```

Ta nám po stisknutí tlačítka `Delete` v panelu nástrojů spustí funkci pro vymazání článku. V ní nejprve načteme do pole `$cid` primární klíče všech vybraných záznamů a připojíme se na databázi. Poté následuje podmínka ke zjištění, zda-li pole obsahovalo nějaké hodnoty:

```
if(count($cid))
    {
    $cids = implode( ',', $cid );
    $query = "DELETE FROM #__clanky WHERE id IN ( $cids )";
    $db->setQuery( $query );
      if (!$db->query())
      {
      echo "<script> alert('".$db->getErrorMsg()."');
      window.history.go(-1); </script>\n";
      }
    }
```

Pokud ano (funkce `count()` vrátila počet proměnných v poli `cid`), použijeme PHP funkci `implode()`. Ta nám z něj udělá řetězec, ve kterém jsou jednotlivé primární klíče odděleny čárkami a ten je pak použit k sestavení dotazu na databázi k vymazání vybraných záznamů [13]. K tomu účelu musíme použít SQL operátor `IN`, díky kterému můžeme zadat více hodnot v klauzuli `WHERE`. Tyto hodnoty od sebe musí být odděleny čárkami, což náš řetězec splňuje, pak už jen položíme dotaz na databázi. Druhá podmínka kontroluje, jestli funkce `query()` proběhla v pořádku. Pokud ne, vypíšeme pomocí JavaScriptu chybu vrácenou funkcí `getErrorMsg()` a načteme předcházející stránku, pokud ano, tak pokračujeme dále v průběhu funkce, kde ještě zbývá odstranit případný nahraný soubor ze serveru. Nejdříve tady načteme `id` příslušného souboru z tabulky `jos_clanky`, abychom pak mohli odstranit správný soubor ze serveru. Pomocí funkce `getDBO()` z třídy `JFactory` se připojíme na databázi. Do proměnné `$query` vložíme dotaz, ve kterém vybere z tabulky `jos_knihovna_soubory` ten řádek, kterému odpovídá `id` načtené z tabulky pro ukládání článků a provede ho. Tím získáme jméno souboru, protože je uloženo v tabulce ve sloupci `soubor_id`. Ten smažeme pomocí PHP funkce `unlink()`, která má jediný parametr a to název souboru [13]. Nakonec použijeme funkci `redirect()` k návratu na hlavní stránku komponenty.

2.6 Naprogramování veřejné části

Administrátorskou část již máme kompletně hotovou, můžeme přidávat, editovat i mazat všechny záznamy v databázi. Podobný úkol nás čeká i tady ovšem s jitými uživatelskými omezeními.

2.6.1 Zobrazení seznamu záznamů

Pokud si naši komponentu otevřeme ve front endu Joomly, zobrazí se nám pouze nadepsaná hlavička (viz. obr.3). Vytvoříme tedy nejdříve funkci, která nám vypíše seznam všech položek v databázi. Otevřeme tedy soubor `clanky.php` v adresáři `components/com_clanky`. Jako první přidáme funkci `require_once()`, která zajistí vložení souboru `clanky.html.php`. Následuje `addIncludePath()` pro načtení třídy `TableClanky` vytvořené v administrátorské části pro možnosti práce s databází. Rozhodovací blok, reprezentovaný příkazem `switch()`, bude mít prozatím jen jednu větev:

```
switch($task)
{ default:
  showPublishedClanky($option);
  break;
}
```

Příkaz v sekci `default` se spustí pokaždé, když bude proměnná `$task` prázdná, tj. vždy při otevření stránky odkazem z menu. Funkce `showPublishedClanky()` pak načte seznam článků k zobrazení:

```
function showPublishedClanky($option)
  {
  $db =& JFactory::getDBO();
  $query = "SELECT * FROM #__clanky WHERE published = '1' AND public = '1'
            ORDER BY title";
  $db->setQuery( $query );
  $rows = $db->loadObjectList();
  if ($db->getErrorNum())
    {
    echo $db->stderr();
    return false;
    }
  HTML_clanky::showClanky($rows, $option);
  }
```

Nejprve proběhne načtení aktuálního spojení s databází pomocí funkce `getDBO()` z třídy `JFactory`. Do proměnné `$query` vkládáme dotaz na databázi. Zobrazeny budou pouze ty položky, které byly uloženy jako publikovatelné a jako veřejné, tzn. mohou si je prohlédnout všichni návštěvníci webu. Pokud má proměnná `public` hodnotu 0, bude zobrazena pouze uživateli, který záznam uložil do databáze, ale o tom až později [6]. Direktiva `ORDER BY` definuje, podle jakých položek má být výstup setříděn, v tomto případě

tedy podle názvu článku. Pak už jen odešleme dotaz na databázi a do proměnné `$rows` uložíme získaná výsledky. Pokud během provádění dotazu nastala nějaká chyba, vypíšeme ji na obrazovku, pokud ne, předáme výsledky k zobrazení. K tomu slouží třída `HTML_clanky`, kterou musíme vytvořit v souboru `clanky.html.php`. Soubor tedy bude obsahovat následující kód:

```
<?php
class HTML_clanky
{
function showClanky($rows, $option)
  { ?><table><?php
  foreach($rows as $row)
  {
  $link = 'index.php?option=' . $option . '&id=' . $row->id . '&task=view';
  echo '<tr><td>
        <a href="' . $link . '">' . $row->title . '</a>
        </td></tr>';
  } ?></table><?php
  } } ?>
```

Na prvním řádku definujeme novou třídu, kterou budeme používat i v jiných funkcí k zobrazení každého našeho výstupu. Následuje samotná funkce pro vypsání záznamů, které parametrem předáváme data získaná z databáze a název komponenty. Cyklus `foreach()` vypisuje odkazy na jednotlivé články tak dlouho, dokud nenarazí na poslední prvek v poli `$rows` (viz. obr.11).

Obr.11: Seznam záznamů.

2.6.2 Zobrazení detailů záznamu

Když nyní klikneme na libovolný článek zobrazí se nám opět stránka s jejich seznamem. Nemáme totiž naprogramovanou žádnou funkci k zobrazení jejich detailů, proto otevřeme soubor `clanky.php` a přidáme následující funkci:

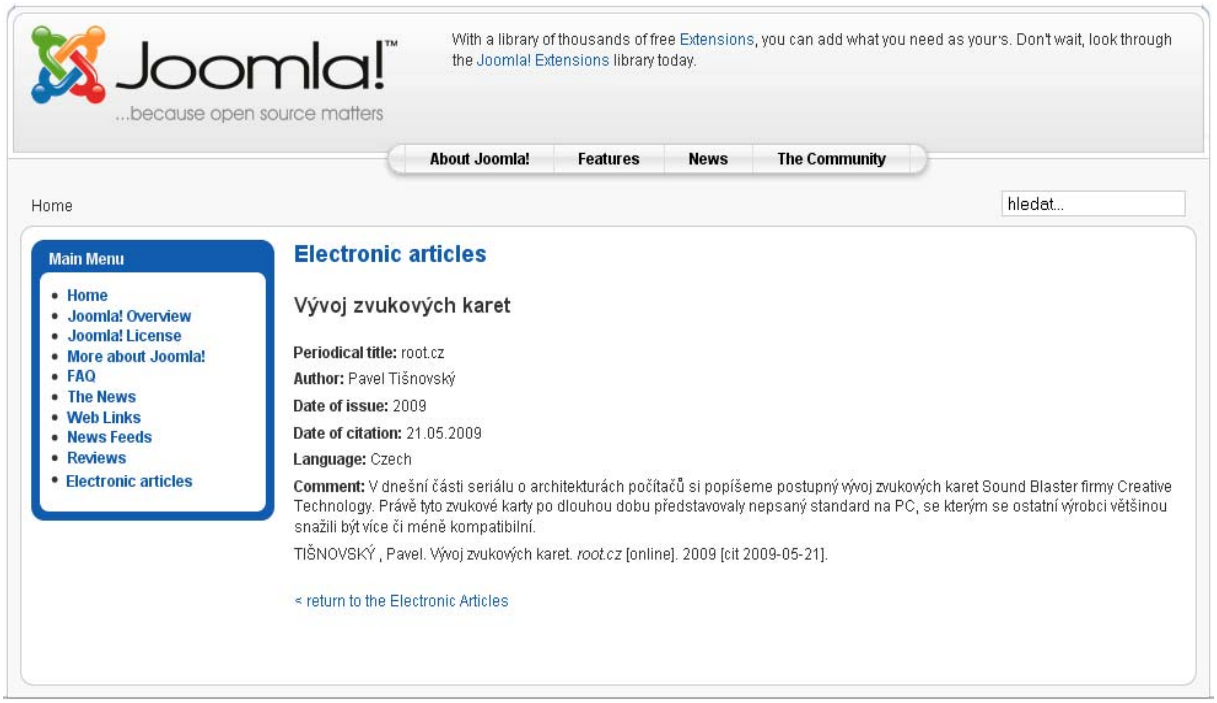
```
function viewClanek($option)
 {$id = JRequest::getVar('id', 0);
  $row =& JTable::getInstance('clanek', 'Table');
  $row->load($id);
  if(!$row->published)
    { JError::raiseError( 404, JText::_( 'Invalid ID provided' ) ); }
  $uploaded_file = $row->uploaded_file;

  $db =& JFactory::getDBO();
  $db->setQuery("SELECT * FROM #__knihovna_soubory WHERE id =
                '$uploaded_file'");
  $rows = $db->loadObjectList();
  foreach($rows as $row2)
    { }
  HTML_clanky::showClanek($row, $option, $row2);
  }
```

V ní nejprve načteme proměnnou `id` pomocí funkce `getVar()`, která zkontroluje, zda-li neobsahuje nějaké nepovolené znaky umožňující útok na naše webové stránky. Protože se nacházíme ve veřejné části systému, je dobré tuto funkci používat pro každou takto získanou proměnnou a zajistit si tak dostatečnou bezpečnost pro naše stránky. Pokud je v ní uložena zakázaná hodnota nebo je proměnná prázdná, je nahrazena nulou a nedojde tak k žádnému dotazu na databázi. Poté získáme instanci naší třídy pro tabulku z administrátorské části a načteme řádek odpovídající hodnotě v `id`. Následuje kontrola proměnné `published`, abychom se ujistili, že záznam můžeme skutečně zobrazit. Pokud ne, vypíšeme pomocí funkce `raiseError()` chybovou hlášku o neexistující stránce [6]. Je to z toho důvodu, aby si uživatelé nemohli sami zadávat do proměnné `id` libovolné hodnoty a zobrazovat si tak nepublikované stránky. Ze stejné tabulky ještě načteme hodnotu primárního klíče z tabulky pro ukládání souborů, připojíme se na ni a získáme název souboru (pokud byl nějaký uložen). Nakonec všechny získané hodnoty předáme ke zpracování v naší třídě `HTML_clanky`. Aby se nám funkce `viewClanek()` spustila, nesmíme zapomenout připsat další větev do příkazu `switch()`:

```
switch($task)
{
  case 'view':
  viewClanek($option);
  break;
```

Nyní se přesuneme do souboru `clanky.html.php`, kde naprogramujeme funkci `showClanek()`, která zajistí vypsání všech údajů na obrazovku. Funkce převezme název komponenty jako parametr a hodnoty z tabulky jako objekt. Pak už následuje vypsání údajů uspořádaných do tabulky. Uvnitř funkce se ještě nachází podmínky pro testování proměnných `uploaded_file` a `comment`, ty se zobrazí pouze pokud je v nich uložena nějaká hodnota, zbytek je povinný. Nakonec se zadaných hodnot vygeneruje bibliografická citace a zobrazí se odkaz pro návrat na hlavní stránku komponenty (viz. obr.12).



Obr.12: Detaily záznamu.

2.6.3 Přidávání, editace a mazání záznamů

Protože běžní uživatelé nemají přístup do administrátorské části, umožníme jim přidávání záznamů z front endu. Po registraci a přihlášení budou moci přidávat, mazat a editovat pouze svoje vlastní záznamy, ne jako v back endu, kde má administrátor přístup ke všem údajům. Upravíme tedy soubor `clanky.php`. Nejdříve přidáme podmínku, která zjistí, jestli je uživatel přihlášený nebo ne. K tomu využijeme funkci Joomly `getUser()` z třídy `JFactory`, která vrací odkaz na objekt `user` a to pokud je uživatel přihlášen.

Podmínkou `if` zkontrolujeme, zda-li vlastnost `name` z objektu `user` obsahuje nějakou hodnotu. Pokud ne, bude uživateli umožněno používat pouze předešlé dvě funkce (zobrazení seznamu záznamů a detailu záznamu), pokud ano, budou mu zpřístupněny i ostatní pro správu záznamů. Do „kladné“ větve podmínky tedy přidáme další příkaz `switch()`, který bude obsahovat prozatím pouze direktivu `default`:

```
default:
showMenu($option);
showPublishedClankyLog($option);
break;
```

Funkce `showMenu()` zkontroluje zda-li je uživatel přihlášen a výsledek předá ke zpracování stejnojmenné funkci v třídě `HTML_clanky`. Pokud byl výsledek kontroly kladný, funkce zobrazí odkaz na stránku pro přidávání nových záznamů (viz. obr.13), pokud ne, nevypíše se na obrazovku nic.

Obr.13: Administrace uživatelské části.

Funkce `showPublishedClankyLog()` slouží pro načtení záznamů přidaných přihlášeným uživatelem. Nejdříve načteme do proměnné `$name` jméno uživatele pomocí funkce `getUser()`, připojíme se na databázi (funkce `getDBO()`), a položíme dotaz na databázi. V něm vybereme z tabulky `jos_clanky` pouze ty záznamy, které přidal právě přihlášený uživatel (hodnota ve sloupci `user` tabulky odpovídá proměnné `$name`). Pokud dotaz vrátil nějakou chybu, je vypsána na obrazovku a uživatel je vrácen na předcházející

stránku, pokud ne, jsou hodnoty získané z tabulky předány k vypsání na obrazovku. K tomu slouží funkce `showClankyLog()` v souboru `clanky.html.php`. Uživateli je vypsán seznam jeho záznamů s možností jejich mazání a editací (viz. obr.13). Protože zde však nemáme panel nástrojů jako v administrátorské část, musíme to provést pomocí předávání odkazem.

Před samotným naprogramováním funkce pro uložení dat musíme stejně jako v administrátorské části nejdříve vytvořit formulář, do kterého budeme data zadávat. Do příkazu `switch()` zmíněného výše přidáme další větev:

```
case 'edit':
case 'add':
editClanky( $option, $id );
break;
```

Jak je vidět, použijeme úplně stejnou funkci jako v back endové části, takže se s ní nemusíme znovu složitě programovat. Nemůžeme ji ovšem použít přesně tak jak je, je třeba udělat několik drobných úprav. Jak je ze zápisu funkce vidět, předáváme jí ještě jeden parametr navíc a to primární klíč, ten ovšem využijeme až při editaci záznamů, na který použijeme stejnou funkci. Nyní si otevřeme soubor `clanky.html.php` kam vložíme stejnou funkci `editClanky()` z administrátorské části, opět s několika úpravami. První z nich je kontrola vložených údajů. Aby nedocházelo k vkládání prázdných záznamů a zahlcování databáze, přidáme jednoduchou funkci napsanou v JavaScriptu, která zkontroluje vyplnění polí:

```
<script type="text/javascript">
  function kontrFormular() {
   if (document.mujFormular.name.value == "") {
    document.mujFormular.name.focus();
    alert("Add author's name.");
    return false;
   }
  return true; }
 </script>
```

V podmínce `if` kontrolujeme, zda-li je prvek `name` z formuláře `mujFormular` prázdný. Pokud ano, funkce `focus()` nás do onoho prvku přepne, `alert()` vypíše chybovou hlášku a funkce zastaví odeslání formuláře. Pokud je vše v pořádku, skript proběhne a data jsou odeslána k uložení do databáze. Samotnému formuláři musíme změnit jméno na mujFormular a přidat do něj událost `onSubmit="return kontrFormular()"`. Ta nám zajistí, že po odeslání formuláře tlačítkem typu `SUBMIT`

proběhne kontrola jeho polí. Prvek pro zadávání jména uživatele změníme na typ `hidden` a automaticky do něj vložíme jméno přihlášeného uživatele, získané funkcí `getUser()`. Tím zabráníme přidávání článků pod cizím jménem. Na konec formuláře ještě doplníme prvky `input`. Jeden typu `hidden` pro vložení hodnoty potřebné ke spuštění funkce pro ukládání záznamů do proměnné `task`, druhý typu `submit` pro odeslání formuláře a třetí typu `button`, pro zrušení zadávání a návratu na předcházející stránku.

A nyní již k samotné funkci pro uložení záznamů do databáze. Do bloku `switch()` tedy přidáme další větev:

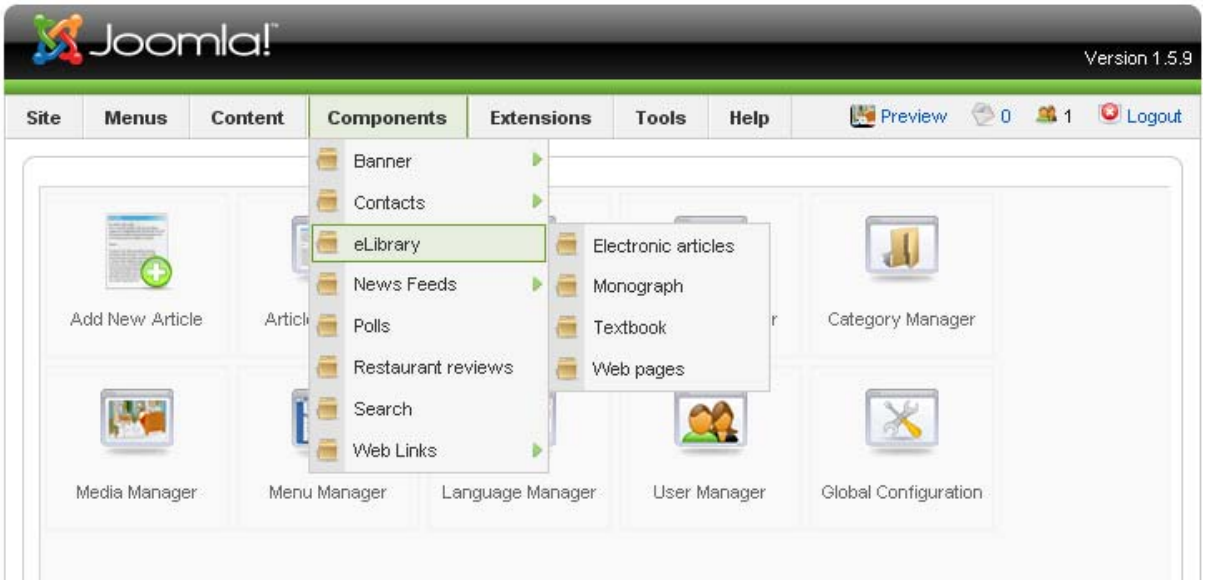
```
case 'pridej':
uploader($dir_upload);
pridejClanky($option, $uploaded_file);
break;
```

Funkce `uploader()` slouží pro nahrávání souborů a je úplně stejné jako v administrátorské části. Funkce `pridejClanky()` je také stejná (`saveClanky()` v back endu), jen v ní není příkaz `switch()`, protože zde nemáme nadefinované tlačítko `apply`.

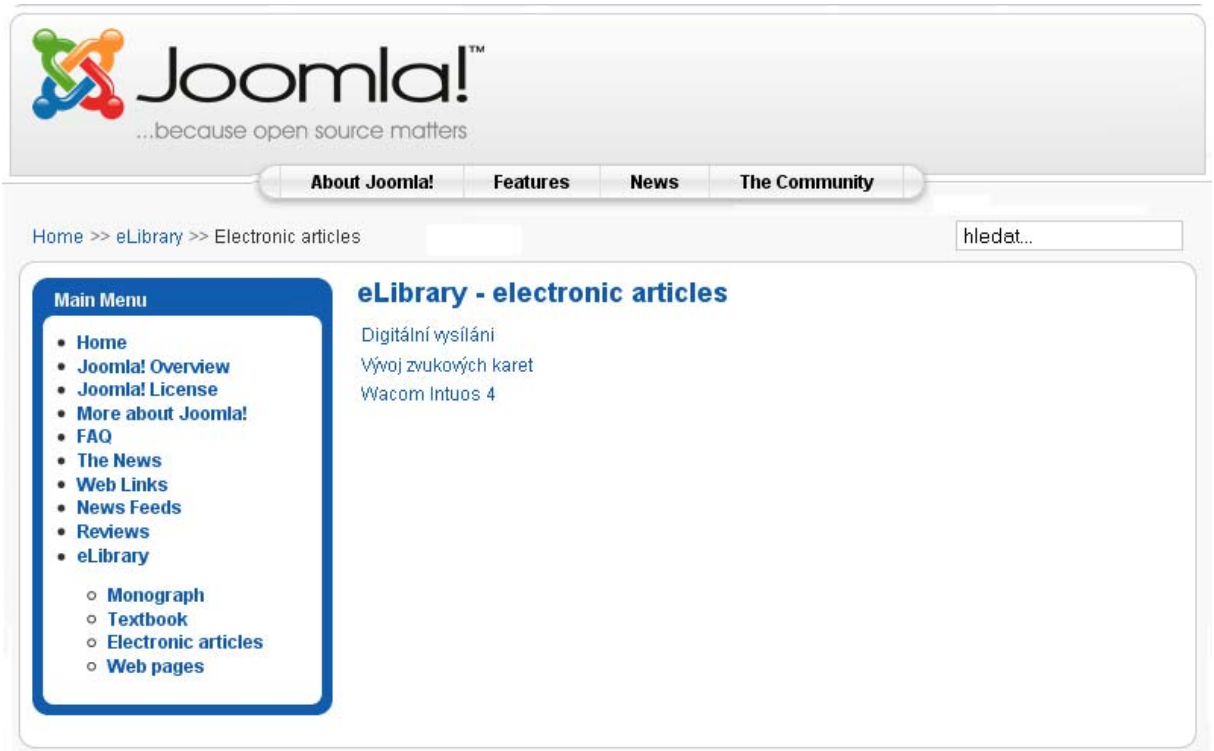
Funkci pro mazání, použitou ze souboru `admin.clanky.php`, poupravíme tak, že odstraníme proměnnou `$cid` reprezentující pole primárních klíčů pro mazání více záznamů najednou, bude zde totiž možné mazat pouze jeden řádek současně. Přidáme funkci `getVar()` pro získání hodnoty `id` mazaného článku a upravíme dotaz na databázi, kde tuto hodnotu použijeme, zbytek už zůstane stejný.

2.7 Kompletace aplikace

Nyní, když můžeme přidávat, mazat a editovat články v systému, máme komponentu hotovou. Podobným způsobem vytvoříme zbývající části a nakonec vše propojíme v menu. Přihlásíme se tedy do PhPMyAdmina a do tabulky `jos_component` přidáme další záznam. Vyplníme zde ovšem pouze sloupce `name` (eLibrary), `admin_menu_img` (stejná hodnota jako u komponent) a `enabled` (hodnota 1). Drobnou změnu provedeme i u našich komponent, kam do sloupce `parent` vložíme `id` knihovny. Ve výsledku pak dostaneme menu jako na obrázku 14. Něco podobného provede i pro veřejnou část. Přihlásíme se tedy do back endu, kde vybereme Menus ->Main Menu, zde klikneme na New a pak na eLibrary. Na následující obrazovce vyplníme název komponenty, její URL a seznamu Parent Item zvolíme rovněž eLibrary. Po uložení máme ve veřejné části menu jako na obrázku 15.



Obr.14: Kompletní menu back endové části.



Obr.15: Kompletní menu front endové části.

3 ZÁVĚR

Bakalářská práce je zaměřena na open source redakční systém Joomla. Pro snazší orientaci je v úvodu uvedena struktura systému (jeho rozdělení na uživatelskou a administrátorskou část a jejich funkce) a možnosti jeho úprav a rozšiřování.

Hlavní část práce je praktického charakteru. Popisuje postup vytvoření komponenty pro elektronickou knihovnu. V první řadě jsem se zaměřil na registraci komponenty do systému a vytvoření panelu nástrojů, protože by jinak bylo složité s ní pracovat a ověřovat funkčnost naprogramovaných funkcí. Poté následovalo vytvoření administrátorské části. Ta se skládá z několika funkcí, z nichž hlavní slouží pro přidávání, mazání a editaci záznamů s možností nahrávání souborů. Funkce jsou rozděleny do dvou souborů a to z toho důvodu, že Joomla odděluje funkční část (skripty pro běh programu) od vizuální (skripty pro výpis výsledků na obrazovku). Pro veřejnou část byly použity funkce z back endu, které však bylo nutné nejdříve upravit, protože zde není možné využít všech prvků Joomly jako v administrátorské části a také bylo nutné zajistit, aby se uživatelé dostali pouze ke svým záznamům a nemohli tak manipulovat s kompletním obsahem databáze.

Ze získaných informací a zkušeností s tímto systémem můžu říct, že Joomla je opravdu kvalitním CMS, který poskytuje spoustu funkcí a nepřeberné množství rozšíření. Nejen díky tomu, ale také díky své jednoduchosti v ovládání a jazykovým variacím jeho obliba u uživatelů na celém světě každoročně roste.

Celá aplikace byla testována na operačním systému Windows XP, Joomla 1.5.9, webovém serveru Apache 2.2.0, databázi MySQL 5.0.18 a PHP 5.1.2. Měla by být kompatibilní i se staršími verzemi, avšak u PHP pouze od verze 4 a vyšší (u starších verzí chybí podpora některých funkcí).

LITERATURA A ZDROJE

Elektronické zdroje:

[1] *Joomla!* [online]. 2009 [cit. 2009-01-08]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Joomla!>.

[2] LUKÁŠ, Jiří. *Co je to redakční systém?* [online]. 2005 [cit. 2005-05-05]. Dostupný z WWW: <http://www.supersvet.cz/view.php?nazevclanku=co-je-to-redakcni-system&cisloclanku=2005050501>.

[3] *Komponenta* [online]. 2004-2009 [cit. 2004-10-15]. Dostupný z WWW: <http://www.joomlaportal.cz/content/view/25/43/>.

[4] *Modul* [online]. 2004-2009 [cit. 2004-10-15]. Dostupný z WWW: <http://www.joomlaportal.cz/content/view/25/43/>.

[5] *Šablona* [online]. 2004-2009 [cit. 2004-10-15]. Dostupný z WWW: <http://www.joomlaportal.cz/content/view/25/43/>.

[6] LEBLANC, Joseph. *Learning Joomla! 1.5 Extension Development : Creating Modules, Components, and Plug-Ins with PHP*. [s.l.] : Packt Publishing, 2007. 176 s.

[7] GRAF, Hagen. *Building Websites with Joomla! 1.5*. [s.l.] : [s.n.], 2008. 363 s.

[8] VÍT, Svatopluk. *Moderní historie – Mambo ve 21. století* [online]. 2006 [cit. 2006-04-09]. Dostupný z WWW: <http://www.joomlaportal.cz/content/view/180/2/>.

[9] *Inside Joomla MVC Part 3* [online]. 2009 [cit. 2009-03-20]. Dostupný z WWW: <http://xtremax.com/blog/2009/03/inside-joomla-mvc-part-3/>.

[10] LAHVIČKA, Jiří. *PHP - práce s řetězci v PHP podruhé* [online]. 2000 [cit. 2000-08-28]. Dostupný z WWW: <http://interval.cz/clanky/php-prace-s-retezci-v-php-podruhe/>.

[11] *Array Functions* [online]. 2001-2009 [cit. 2009-05-22]. Dostupný z WWW: <http://cz2.php.net/in_array>.

[12] *Control Structures* [online]. 2001-2009 [cit. 2009-05-22]. Dostupný z WWW: <http://cz2.php.net/manual/en/control-structures.foreach.php>.

[13] *PHP Manual* [online]. 2001-2009 [cit. 2009-05-22]. Dostupný z WWW: <http://cz2.php.net/manual/en/index.php>.

SEZNAM PŘÍLOH

PŘÍLOHA 1

Obsah přiloženého CD:

Adresář	Obsah adresáře
Bakalářská práce	Bakalářská práce ve formátu doc i pdf + zadání ve fromátu pdf
Elektronická knihovna	Podadresář administrátor se zdrojovými kódy pro back end, podadresář components se zdrojovými kódy pro front end, soubor tabulky.txt se strukturou databázových tabulek