Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Rozšíření systému OpenCms pro KIV

Plzeň, 2009 Stanislav Skalický

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 16. 5. 2009, Stanislav Skalický

Poděkování

Rád bych poděkoval Ing. Přemyslu Bradovi, MSc., Ph.D. za věnovaný čas, podnětné připomínky a trpělivou spolupráci, Ing. Josefu Krupičkovi za čas věnovaný konzultacím některých bodů zadání práce.

Abstract

OpenCms extensions for KIV

The main goal of this diploma thesis is to analyze the details of OpenCms system, which is used for web content management on the official web sites of the Department of Computer Science and Engineering on University of West Bohemia in Pilsen. This study should result in a summary of some suggestions and requirements on possible extensions of OpenCms core functions to improve the usability of the system workplace.

Implementation part of this work consists of two kinds of modules. In the first part some of the proposals for OpenCms core extension will be realized. The second part involves the analysis, design and implementation of new OpenCms modules, which will provide some advanced features for the web sites of the department.

Obsah

1 Úvod

Na katedře KIV již několik let probíhají práce související s přestavbou katedrálního webu. Primárním cílem této přestavby bylo vytvořit snadno udržovatelný a rozšiřitelný web, který bude navíc technologicky a uživatelsky vyspělý. Bylo nutné jej tedy postavit na kvalitním systému pro správu webového obsahu. K tomuto účelu byl v rámci diplomové práce Josefa Krupičky [4] vybrán a nakonfigurován systém pro správu obsahu založený na technologii Java a XML, jehož název je OpenCms. Následně proběhly první pokusy o nasazení systému do provozu a do jeho dalšího vývoje bylo zapojeno několik studentů a pracovníků katedry v čele s Ing. Přemyslem Bradou, MSc., Ph.D.

V současné době jsou na katedře v provozu dvě verze webových stránek postavených na sytému OpenCms. Mimo oficiálního webu katedry je tady ještě testovací verze stránek, která poskytuje prostor pro vývoj nových modulů do OpenCms. Oficiální stránky již zahrnují jádro původního webu i s několika doplňujícími moduly. Moduly původního webu, které ještě nejsou nasazeny na oficiální verzi stránek, se pak nacházejí v závěrečné fázi vývoje a běží v testovacím provozu. Důraz je kladen také na další rozvoj stránek a implementaci nových funkčních modulů.

1.1 Cíl práce

Přestože za systémem OpenCms stojí rozsáhlá aktivní komunita, a ten se tedy neustále vyvíjí, vznikla za dobu jeho užívání na katedře KIV řada požadavků a návrhů na vylepšení jeho funkčnosti spojených s konkrétními potřebami katedry.

Cílem této práce je prostudovat implementační detaily OpenCms a sumarizovat návrhy a požadavky na možná rozšíření jádra systému ze strany komunity OpenCms, ze strany katedry KIV a z výsledků dalšího průzkumu. Tyto návrhy bude třeba porovnat mezi sebou a také s budoucím plánem vývoje systému ze strany jeho tvůrců. Na základě výsledků porovnání budou vybrané náměty implementovány v rámci této práce, ostatní náměty pak budou doporučeny pro další rozšíření jádra OpenCms.

V druhé fázi této práce bude hlavním úkolem analyzovat, navrhnout a implementovat OpenCms moduly, které budou doplňovat webové stránky katedry o další pokročilé funkčnosti.

1.2 Přehled kapitol

Teoretický rozbor před samotnou implementací začíná kapitolou 2, ve které jsem se snažil analyzovat požadavky, které jsou v současných webových prezentacích kladeny na systémy pro správu obsahu. V této kapitole dále rozebírám zavedené postupy správy obsahu a nástroje, které se k tomu vyžívají. Většina této kapitoly je založena na zřejmě nejrozsáhlejší publikaci, věnující se obecně správě obsahu nejen na webu, kterou napsal Bob Boiko[6].

Ve třetí kapitole vás seznámím s implementačními detaily a klíčovými prvky OpenCms a podělím se o zkušenosti s tímto systémem, které jsem získal během tří let práce s ním.

V kapitole 4 se budu věnovat komunitě, která stojí za vývojem OpenCms, a ve spojitosti s ní také námětům na vylepšení OpenCms. Výstupem pak budou návrhy na rozšíření, které budou v rámci této práce implementovány.

Kapitola 5 se věnuje právě popisu rozšíření jádra systému OpenCms o nové funkčnosti, jejichž úkolem je zefektivnit práci s některými částmi uživatelského prostředí OpenCms.

V šesté kapitole budou popsány OpenCms moduly, které doplňují katedrální web o pokročilé funkčnosti.

Na závěr zhodnotím výsledky této práce, do jaké míry byly splněny její cíle, a nastíním, kudy by se mohl ubírat další vývoj nových součástí webu katedry.

2 Systémy pro správu obsahu

Systémy pro správu obsahu, neboli content management systems (dále jen CMS), jsou softwarové aplikace, které podporují vytváření, správu, distribuci a prezentaci obsahu.

Obsahem v tomto případě nemyslím pouze kus textu. Na obsah může být z definice CMS nahlíženo jako na objekt, který má formát, strukturu a funkcionalitu. Funkcionalita představuje něco, co vykonáme (např. odešleme požadavek, odpověď na požadavek). Obsah také vždy souvisí s kontextem, který ještě více prohlubuje jeho význam.

Správa obsahu je tedy o získání kontroly nad vytvářením, zařazováním a funkcionalitou informací. Je to proces pro shromažďování, správu a publikování informací.

Systémy pro správu obsahu se v češtině často označují jako redakční, nebo publikační systémy. Tyto pojmy však nejsou úplně přesné, CMS je totiž poměrně obecný výraz. Systémy pro správu obsahu se dále mohou dělit do několika kategorií, které konkretizují působnost systémů. Z těchto kategorií stojí za zmínku následující:

- **Document management system (DMS)** – Systémy určené pro správu dokumentů v nějaké organizaci. Primárním cílem tedy není publikování informací na internetu, ale spíše jejich správa v intranetu organizace [1]. DMS by měl organizaci poskytovat nástroje pro vytváření dokumentů a určování jeho toku v rámci organizace. Měl by tedy obsahovat centralizované úložiště pro dokumenty a nějaké workflow zapouzdřující pravidla a metadata [2]. DMS používá pro správu často „tlusté klienty".
- **Enterprise content management system (ECMS)** – Systémy, které v sobě spojují více nezávislých zdrojů dat, jež pak zobrazují [1]. ECMS obsahuje hlavní CMS, který disponuje vedlejšími schopnostmi umožňujícími správu širšího spektra informací v nějaké organizaci. Mimo jiné ECMS může zahrnovat DMS, WCMS a další konkrétní systémy [2].
- **Web content management system (WCMS)** – Systémy, které jsou vystavěny výhradně na webových technologiích. Pro správu obsahu využívají u většiny případů webové rozhraní, takže autoři a návštěvníci potřebují na své straně pouze připojení k internetu a webový prohlížeč [3]. WCMS se typicky zaměřují na publikaci informací

na internetu. WCMS je dnes ze všech konkrétních systémů daleko nejrozšířenější, o čemž vypovídá fakt, že pokud se řekne CMS, je tím myšlen právě WCMS.

Dále se v této práci budu zabývat webovými systémy pro správu obsahu a budu je nadále označovat pouze jako CMS.

2.1 Požadavky na CMS

CMS je velice obecný pojem a v zásadě bude jeho specifikace a složitost v mnoha případech velice odlišná. Požadavky se budou odvíjet od jeho primárního cíle působnosti. Jiné požadavky budou kladeny na CMS určené pro správu blogu, jiné pro správu diskusního fóra, elektronického obchodu, nebo rozsáhlého portálu. Také bude při jejich tvorbě záležet na tom, zda Stakeholderem je fyzická osoba, nebo velká firma.

V této kapitole tedy zmíním spíše všeobecné cíle systémů pro správu obsahu a požadavky z hlediska uživatelů angažovaných v CMS.

2.1.1 Cíle CMS

Na systémech pro správu obsahu dnes prakticky závisí úspěch internetových stránek, protože CMS pokrývá kompletní životní cyklus obsahu na stránkách, a to od poskytování nástrojů pro vytváření obsahu, přes publikování, až k archivaci informací.

Jaké výhody by tedy měly tyto systémy přinést uživateli k publikování informací na internetu? Odpovědí je následující seznam hlavních cílů, které jsou dnes již standardem kvalitních CMS. Cílů však může být mnohem více, záleží na tom, jaký typ stránek chceme provozovat.

- **Zpřístupnit vytváření obsahu** - CMS by měl zpřehlednit, zrychlit a zjednodušit operace, které se týkají vytváření obsahu, zvýšit flexibilitu stránek a podporovat informační růst. Zejména je třeba umožnit editaci obsahu i uživatelům bez technických znalostí programovacích jazyků [4].
- **Zavést organizaci a správu obsahu** – obecně by CMS měl snížit náklady na údržbu obsahu. Z údržby mohu pak vyseparovat celou řadu konkrétnějších cílů:
    - CMS by měl zvýšit kvalitu poskytovaných informací, která závisí na přesnosti, úplnosti a aktuálnosti informací.

  - Odstranit duplicitu informací, protože duplicita zvyšuje náklady na údržbu a chybovost. Informace by tedy měly být v systému uloženy pouze jednou a do více stránek vkládány pomocí referencí.
  - Zvýšit výkonnost personálu. CMS by měl poskytovat rozhraní pro vymezení konkrétních úkolů pro personál tak, aby se každý v našem systému staral jen o to, co náleží jeho roli. Např. obsah webu by měli psát ti, kdo mají informace, jiní uživatelé budou mít na starosti zase vytváření šablon a jiní budou vytvářet grafiku do těchto šablon.
  - Uložené informace by měly být v systému snadno dohledatelné s pomocí integrovaných filtrů, vyhledávačů apod.
- **Snížit náklady na publikaci obsahu** – CMS by měl zefektivnit a zautomatizovat procesy publikace informací. Způsob jejich prezentace by měl být snadno změnitelný. Také by měl přenést odpovědnost za publikace na jejich autory zvýšením transparentnosti publikování. Zúčastněnému personálu také pomohou různé online dostupné manuály a tutoriály popisující postupy pro práci s CMS.
- **Zvýšení návštěvnosti webových stránek** - je konečným cílem, který vyplývá ze všech předchozích [5].

2.1.2 Uživatelé v CMS

Jedním z hlavních cílů systémů pro správu obsahu je též umožnit spolupráci více uživatelů. Již samotná idea CMS vybízí ke spolupráci různých typů uživatelů. CMS by měl tedy umožnit editaci informací uživatelům technologicky neznalým a zároveň by měl poskytovat rozhraní pro někoho, kdo bude tyto uživatele spravovat, poskytovat jim šablony, obsluhovat jejich požadavky atd. V tomto případě dnes již ani nezáleží na velikosti CMS. Každý by měl poskytovat rozhraní pro interakci více typů uživatelů, aby mezi ně bylo možné procesy v rámci CMS vhodně rozdělit. Každý uživatel pak bude mít na starosti jen procesy, které náleží jeho roli v systému a podle toho bude mít i nastavená přístupová práva a odpovědnosti.

Tato kapitola dále popisuje uživatelské role CMS. Vysvětluje, proč vlastně určité role CMS potřebuje a co ty od něj očekávají. Uživatelských rolí by se samozřejmě dalo vymyslet ještě více na základě konkrétní specializace CMS, popř. na základě potřeb organizace, kde je

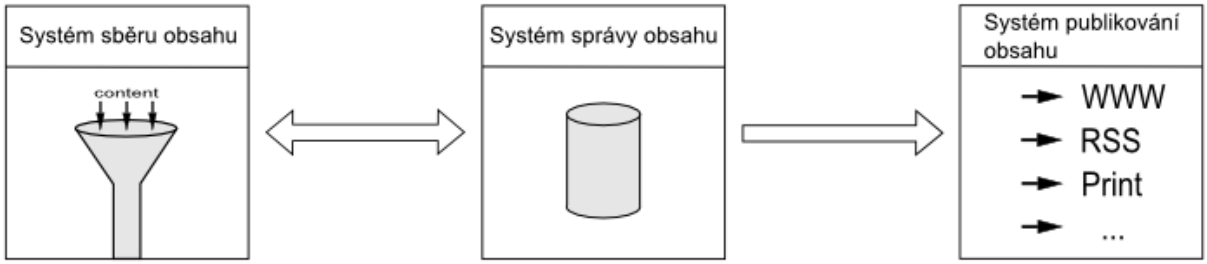
CMS nasazen. Já se však budu věnovat nejdůležitějším rolím, které se vyskytují prakticky ve všech CMS:

- **Administrátor** – správce CMS, který jej nainstaloval na server. Má za úkol monitorovat provoz na serveru a inicializační low-level konfiguraci CMS. Tento uživatel by měl mít přístup ke všem dostupným funkcím CMS, i když většinu z nich vůbec nebude využívat (na to jsou v systému jiní uživatelé). Administrátor je nejvíce spokojen, když jej project manager delší dobu nekontaktuje, protože to pro rutinní provoz není potřeba [3].
- **Projektový manažer** – šéfredaktor CMS, který se stará o personál, zadává úkoly, má poslední slovo při publikaci informací. Dohlíží na všechny, nebo pouze vybrané projekty v rámci CMS.
- **Personální manažer** – najímá, propouští a řídí uživatele, kteří jsou hierarchicky pod ním. CMS by měl poskytnout tomuto uživateli rozhraní pro správu uživatelů, uživatelských skupin i uživatelských rolí.
- **Systémový vývojář** – uživatel, který zná implementační detaily CMS. Jeho úkolem je vyvíjet nové funkčnosti pro systém ve formě modulů a v případě potřeby upravovat funkci jádra CMS, pokud je mu to umožněno. Výhodou CMS z hlediska tohoto uživatele je výrazné oddělení prezentační, aplikační a datové části CMS.
- **Grafik a vývojář šablon** - využívá šablonový systém CMS a vytváří šablony z různě dostupných komponent a elementů systému. Stará se i o design publikací.
- **Editor** – Jeho hlavním úkolem je vkládat nové informace do CMS. Je to uživatel, na kterého jsou kladeny nejmenší nároky z technického hlediska. Proto by mu měl CMS poskytnout vhodné, intuitivní nástroje pro vytváření a editaci textů. Neméně důležitá bude dostupnost různých návodů pro práci s CMS. Skutečně se často může jednat o naprosté laiky v oblasti IT, kteří prostě jen mají informace z jiné oblasti a náhle dostali za úkol je publikovat.
- **Čtenář** – CMS může čtenáři nabízet možnost personalizace webu. Díky tomu může čtenář přizpůsobit vzhled publikací svým potřebám. V nejjednodušším případě může jít o výběr předpřipravených grafických podob publikací. Pokročilejším případem pak může být možnost přemisťovat elementy, nebo je schovávat.

2.2 CMS a jeho kostra

Abych mohl probrat CMS procesy, entity a nástroje podrobně, nejprve si CMS pro přehlednost rozdělím do několika hlavních kategorií.

Jak už bylo řečeno, správa obsahu sestává z procesů pro shromažďování, správu a publikování informací, jak naznačuje Obr-2.1. [1] Tyto procesy jsou pak mezi sebou vázány pomocí workflow.



Obr-2.1 – Kostra CMS

CMS jako systém, který shromažďuje, spravuje a publikuje informace a funkcionalitu, popíšu stručně v následujících podkapitolách. Podrobnější rozbor pak zahrnuje kapitola 2.4.

2.2.1 Systém sběru obsahu

Systém pro sbírání obsahu (Collection system) zahrnuje procesy, které vyžadují především autoři a tvůrci obsahu. Lze najít několik způsobů sběru informací. Můžeme je buď vytvořit, nebo získat z externích zdrojů. V závislosti na zdroji pak můžeme a nemusíme informace konvertovat do nějakého žádoucího formátu (XML, HTML …). Nakonec shromažďujeme informace do systému menšími úpravami, rozčleněním na různé spolu související útržky informací, nebo komponenty. Zároveň v této fázi přidáváme k obsahu metadata (některá jsou přidána systémem automaticky, jiná musíme zadat manuálně).

[1] Do kostry CMS bych mohl zahrnout ještě fáze prezentace a fázi nasazení (viz [7]), ale rozhodl jsem se ponechat kostru v původním stavu ze dvou důvodů: Fázi prezentace považuji za součást fáze publikování informací. Fáze nasazení, která zahrnuje procesy jako zaučení editorů, dokumentace postupů, popis používaných konvencí apod., se zase nehodí do této kostry, protože se jedná spíše o fázi vývoje CMS jako softwarového produktu.

2.2.2 Systém správy obsahu

Základem všech CMS by mělo být centrální úložiště obsahu podporované řadou nástrojů pro manipulaci a správu obsahu spolu s jeho administrativními daty (např. data o systémových uživatelích). Systém správy obsahu je zodpovědný za dlouhodobé uchovávání informací v podobě obsahových komponent a za řadu dalších zdrojů v tomto centrálním úložišti. Jeho úkolem je informovat uživatele o detailech obsahu, druzích komponent v systému, obsahových typech, v jakém stádiu životního cyklu se nacházejí. Dále by měl poskytovat informace o tom, jaké komponenty využíváme při publikování, jaké jsou nevyužité, kdo má přístup k jakému obsahu, kdo do systému přispívá nejvíce apod.

2.2.3 Systém publikování obsahu

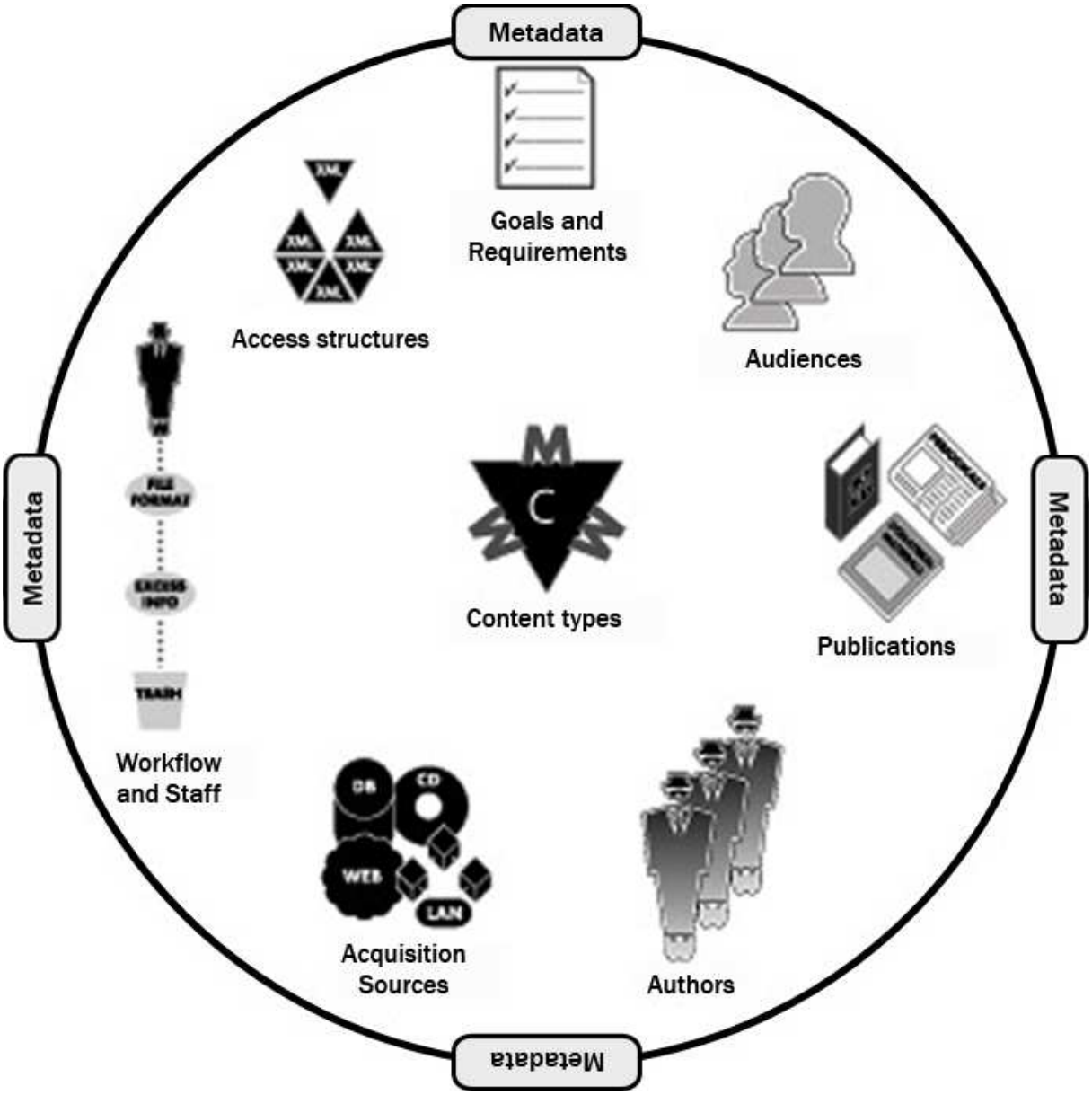
Systém publikování má za úkol vzít si obsah uložený v centrálním úložišti a vygenerovat z něj publikace, které budou prezentovány zájemcům o tyto informace. Publikace mohou být ve formě webových stránek, tisknutelných dokumentů, email newsletters atd. Skládají se z vhodně uspořádaných komponent, funkcionality, standardních okolních informací (banner, patička atd.) a navigace.

Samozřejmostí by mělo být, že publikace (většinou webové stránky) splňují určité předepsané standardy a pravidla přístupnosti.

2.3 Postupy správy dat – Jaké entity hrají roli v CMS

V této kapitole bude popsán logický design CMS, tzn. jaké prvky a postupy by měl obsahovat buď při jeho navrhování, nebo při výběru již hotového řešení. CMS zde tedy budu popisovat ještě z jiného pohledu, než v kapitole 2.2. Tento pohled by měl však prohloubit porozumění toho, o co vlastně v CMS běží.

Skrze systémy sbírání, správy a publikování obsahu je v CMS vytvořeno nepřímé spojení mezi autorem a publikací. Tento systém dovoluje práci autorů kombinovat s dalšími zdrojovými materiály, čímž vzniknou publikace různého druhu. Proto je možné chápat CMS jako sadu významných hráčů, nebo entit, které všechny na sebe vzájemně působí, a tím vytvoří "žijící systém". CMS by měl dále zahrnovat vyváženou interakci mezi působnostmi jednotlivých entit, které tvoří CMS. Bob Boiko ve své knize [6] tuto součinnost mezi entitami nazývá *Wheel of content management* (viz Obr-2.2). Některé z těchto entit budou popsány v následujících podkapitolách.



Obr-2.2 – The wheel of content management[2]

2.3.1 Metadata

Jak již bylo řečeno, CMS by měl poskytovat vyváženou interakci mezi všemi entitami, které v něm působí. Hlavní prvek, který toto zajišťuje, jsou právě metadata. Na Obr-2.2 tvoří lem kruhu, což znázorňuje fakt, že metadata udržují rovnováhu mezi entitami CMS, a tím tvoří stabilní a efektivní systém pro vytváření a používání obsahu.

[2] Autorem obrázku *The wheel of content management* je Bob Boiko a má původ v jeho knize [6]

Metadata, více než cokoli jiného, jsou tím, co celý systém definuje. Jejich nejčastější definice říká, že jsou to data o datech. Z hlediska CMS jsou pak důležitá proto, že přidávají kontext a širší interpretaci datům samotným. Může se jednat např. o datum, stav, komentář, hodnocení obsahu, propojení obsahu s autory a editory apod.

V CMS metadata tvoří sadu jmen, vztahů a strukturu, bez kterých je obsah neuspořádaný, neúplný a neprovázaný. Většinou spadají do několika málo zřetelných kategorií jako *navigační kategorie (*TOC, index), *kategorie správy* (autor, datum vytvoření, poslední editace atd.), *kategorie obsahového typu a interní struktury* (title, abstract, body, atd.) a *kategorie zahrnutí (*přidej media zde, banner támhle, standardní text zde atd.).

Nakonec je třeba si také uvědomit, že stejně jako je tomu u obsahu, i v metadatech by měl být zaveden nějaký systém, a k tomu je nutné poskytnut organizaci a strukturu metadatům. Taktéž je vhodné vytvořit pro účastníky CMS *"metatorial guide"* zavádějící pravidla a štábní kulturu, které budou členové personálu dodržovat ke členění a značkování veškerého obsahu, který se jim dostane na stůl.

2.3.2 Obsahové typy

V srdci každého CMS jsou umístěny jisté kusy informací, které se mohou nazývat obsahovými komponentami[3]. Ty se pak rozdělují do spravovatelných jednotek v několika variacích, a ty nazýváme obsahovými typy.

Na *obr-2.2* se obsahové typy nachází uprostřed kruhu (*Content types*), protože jsou vlastně výsledkem veškeré vědomosti získané o všech ostatních entitách. Jsou také zdroji, ze kterých je poskytována veškerá hodnota systému. Obsahové typy úzce souvisí s další významnou entitou – metadaty. Většina metadat totiž může být uložena právě v elementech obsahových typů, které definujeme. Obsahové typy jsou pak fragmenty informací, nebo funkcionality, které lze zaobalit do metadat.

[3] Komponenty vždy nemusí zaobalovat obsahové typy, ale je vhodné určit, jak budeme v CMS obsahové typy udržovat a kde budou shromážděné. A pokud má CMS modulární architekturu, nabízí se řešení spojení obsahových typů s moduly (komponentami), protože tím definujeme pravidla a zpřehledníme systém přidávání nových obsahových typů do systému.

2.3.2.1 Elementy obsahových typů

Když už byla řeč o elementech[4] obsahových typů, tak ty představují prověřené a používané řešení, jak dále členit obsahové typy ještě do menších kousků informací. Pokud je navíc tento systém elementů dobře navržený, poskytne pak např. tvůrcům šablon další žádoucí možnosti, kdy obecnější šablony budou obsahovat pouze některé prvky obsahového typu, konkrétnější šablony pak všechny elementy obsahového typu.

Element má typy a hodnoty – Content typy rozdělujeme na elementy, aby bylo možné je jednoduše nalézt a udělat obsah uvnitř nich jednodušeji lokalizovatelný a opravitelný. Dále můžeme kategorizovat i elementy na *Body elements* (zahrnují obsah, který právě zobrazujeme v publikacích) a *Managements elements* (tyto elementy vkládají informace, jako datum vytvoření, stav komponenty apod. Tento typ elementů vytváříme pro účel správy komponent).

2.3.2.2 Analýza obsahových typů pro CMS

Různé CMS mohou nabízet širokou škálu již nadefinovaných obsahových typů. Jak už ale bylo řečeno, publikované informace a požadavky na ně často zabíhají do jemných detailů[5], a proto je velice důležité mít v CMS možnost definovat vlastní obsahové typy a modifikovat stávající. Ať už tedy vytváříme nový CMS, nebo vybíráme hotové řešení, je třeba si nejprve určit, jaké obsahové typy budeme potřebovat a jak budou v systému spravovány:

1. Nejprve vytvoříme nějakou širší sadu content types, která bude pokrývat odvětví, pro které bude CMS fungovat.
2. V druhém kole zúžíme obsahové typy jen na ty, které budeme opravdu potřebovat.
3. Je také třeba dbát na to, aby obsahové typy byly v rámci systému jednoznačně identifikovatelné. Přiřadíme jim tedy unikátní ID element, dále název, popis, pozici v systému apod.
4. Nakonec rozhodneme pro každý typ, jak se bude vytvářet, jaké bude obsahovat elementy, jaké elementy jsou běžné napříč všemi typy obsahu apod.

[4] Element ve smyslu XML elementu – element je start tag a end tag a vše, co je mezi nimi. To znamená, že v XML je element základní jednotkou informace a nejmenší jednotkou, kterou umíme najít a pojmenovat. To samé by mělo platit z hlediska CMS – element je nejmenší jednotka obsahu, ke které můžeme zjistit polohu a pojmenovat ji.

[5] Detaily se týkají obzvlášť formátu publikací a mohou záviset na lokalizaci, do které se informace publikují (např. v české lokalizaci rozhodně nebudeme chtít publikovat čas ve formátu 1:24 PM).

Nakonec v tabulce Tab-2.1 uvádím často používané obsahové typy, se kterými se lze setkat na webových stránkách z velkého množství sektorů a odvětví.

Obsahový typ	Popis
Nabídky	Drží informace o všech produktech a službách, které podnik nabízí.
Výroční zprávy	Zprávy publikované s pravidelnou periodou
Události	Události, které společnost plánuje, nebo které již proběhly. Vztahují se ke konkrétnímu datu.
Aktuality	Novinky, které se nemusí vztahovat ke konkrétnímu datu.
Newsletters	Pravidelně distribuované publikace věnující se tématu, které spadá do oblasti zájmu svých odběratelů.
Odkazy	Webové odkazy vedoucí ven z webu společnosti.
Articles	Články, eseje, blogové příspěvky apod.
FAQs	Často kladené otázky.
Zákazníci	Informace o zákaznících, kteří mají co do činění se společností.
Statické stránky	HTML stránky bez dynamických prvků.
Dokumenty	Vytvořené např. prostřednictvím tabulkových kalkulátorů, textových procesorů atd.
Downloads	informace o binárních souborech, které by organizace ráda nabídla ke stažení
Bio	informace o významných osobách v organizaci

Tab-2.1 – Běžné obsahové typy.

2.3.3 Získávání informací

Před tím, než budeme informace publikovat, je nutné je získat z dostupných zdrojů. V této části tedy zanalyzuji možné zdroje informací.

2.3.3.1 Autorství

Autoři jsou lidé, kteří pro nás vytváří originální informace. Nejsou na ně kladeny požadavky na programátorské znalosti, proto by jim měl vybraný systém poskytovat prostředí a omezení zjednodušující jim práci. Při jejich hledání je třeba si zodpovědět následující otázky:

- Zamyslet se, co víme o našich autorech, a zvážit jejich motivaci, schopnosti a kvalifikaci
- Zdroje obsahu – jaké typy obsahu můžeme od autorů získat.
- Pro jaké lokalizace budou autoři vytvářet obsah apod.

2.3.3.2 Import informací

Zatímco autoři píší obsah, získáváním zdrojů zásobujeme CMS obsahem, který již v nějaké formě existuje. Pokud se rozhodneme informace importovat z externích zdrojů, je dobré nejprve vytipovat všechny potencionální zdroje našeho obsahu. Často nebudeme muset zdroje ani hledat, prostě jen vyplynou z požadavků na systém[6]. Otázkou tedy bude, z jakých formátů budeme informace získávat a do jakých obsahových typů CMS bude provedena konverze.

Při analýze získávání informací z externích zdrojů je dobré následovat tyto kroky:

- Rozhodnout, jaké typy obsahu vzniknou z externích informací.
- Jakým způsobem proběhne konverze do obsahových typů systému – měli bychom zvážit, zda se vůbec vyplatí informace importovat, pokud bude konverze náročná.
- Zjistit, jestli máme k dispozici nástroje pro konverzi, popř. jak složité je bude vytvořit.
- Zjistit, kolik obsahu vlastně budeme potřebovat importovat a kolik obsahu je zdroj schopný poskytnout a přizpůsobit se tomu.

Samozřejmostí bude opět zajištění označkování importovaného obsahu podle konvencí v systému. Bude také nutné zavést pravidla na přiřazení metadat k importovanému obsahu. Také bychom si měli uvědomit, že kvalita získaných informací bude různá, proto je dobré zavést i nějaké statistiky a hodnocení zdrojů.

2.3.4 Publikace

Cílem CMS je vytvořit publikace, které návštěvníci od našeho systému požadují. Publikace jsou vlastně tím konečným produktem, který vzejde z CMS.

2.3.4.1 Návštěvníci

Abychom zjistili, jaké publikace od nás mohou návštěvníci požadovat, je třeba nejprve analyzovat samotné potencionální návštěvníky našeho webu. Jedná se vždy o nějakou skupinu lidí s podobnými vlastnostmi, které musíme brát v potaz již při návrhu, či výběru CMS, a to hned z několika úhlů pohledu, včetně marketingu, softwarového vývoje, nebo psaní článků.

[6] Např. v systémech pro státní správu bude nutné publikovat povinně zveřejňované informace, jejichž jediný zdroj bude např. výstup (dokument stálého formátu) z nějaké schůze apod.

Uživatele našeho systému je třeba si na začátku rozčlenit do skupin tak, abychom jim byli na základě tohoto členění schopni doručovat co nejlepší obsah.

Jde o to rozpoznat, kdo do našeho okruhu uživatelů patří a kdo už ne. Můžeme např. sledovat charaktery a rysy lidí a najít tak sadu znaků, která naše uživatele vyznačuje. Tím si vytvoříme jakási metadata o návštěvnících a ta mohou zahrnovat např. následující atributy:

- název,
- věková kategorie,
- zájmy,
- pohlaví,
- často prohlížené stránky apod.

Na základě těchto metadat o uživatelích lze zvolit vhodnou strategii vytváření publikací a navrhnout systém personalizace našeho webu.

2.3.4.2 Analýza publikací

Publikování je vydávání informací. Publikace je tedy informace, kterou vydávají publikátoři. Může se jednat o statické stránky, tisknutelné sestavy, webové syndikace obsahu (RSS), PDF export a další.

Ve větších systémech je tedy záhodno vytvořit tým publikátorů, který bude zodpovědný za vytváření, testování a vydávání publikací. Tito publikátoři jsou pak jen okrajově i samotnými autory informací. Jejich úkolem je spíše rozhodnout, jaký má publikace účel a jak je tento účel svázán s cíli, které chceme se systémem dosáhnout.

Co se týká formy samotných publikací, opět je dobré si nejprve analyzovat atributy a metadata, které budou publikace obsahovat:

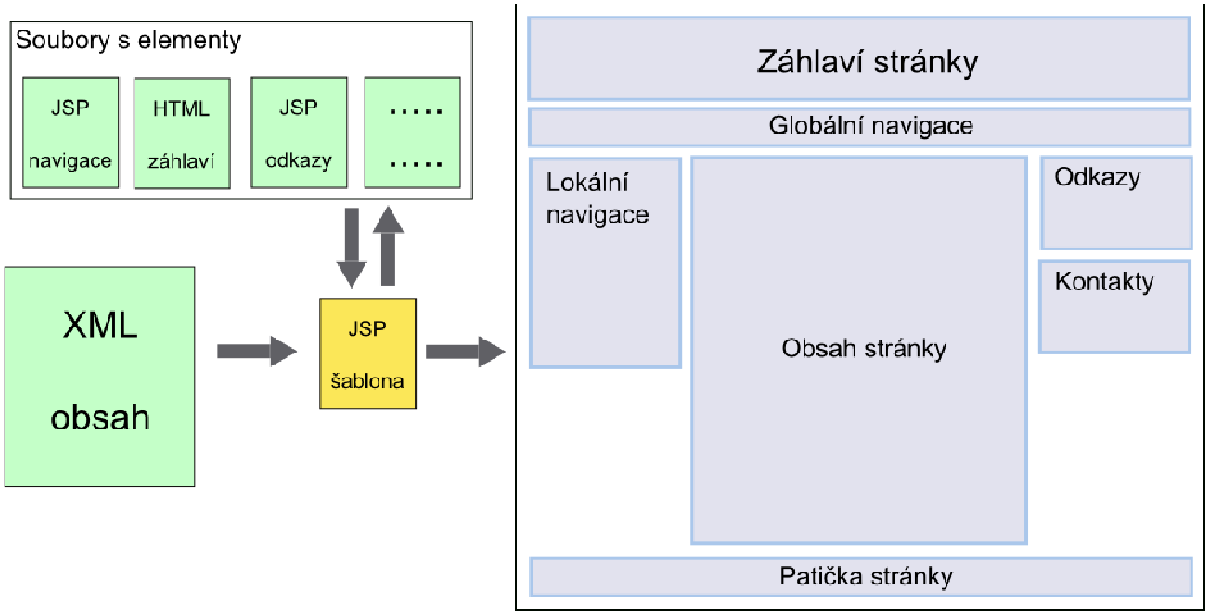
- základní atributy publikací – jméno, audience, publikátoři, autoři, důležitost,
- případy užití pro publikace,
- formát,
- zprávy každé audienci publikace,
- strukturu navigace pro publikaci,
- cykly publikace – zahrnuje velikost, verzování, frekvenci updatu,
- lokalizace.

Publikátor na základě těchto metadat může rozhodnout o formě a místu (URL) publikace.

2.3.5 Šablony

Šablony jsou tím, co v CMS vytváří publikace. Šablony nás ušetří od neustálého rozmísťování obsahu do jednotlivých částí a formy, protože umožňují automaticky vytvářet části publikace. Šablona je funkcionalita, kterou vytvoříme pouze jednou pro rozložení a formát obsahu na publikaci, a poté ji můžeme používat pro libovolný obsah opakovaně. Tvoří spojení mezi obsahem uvnitř CMS a publikacemi zveřejněnými na webu. Důležitým rysem šablon je jejich schopnost udržovat obsah organizovaný a zamezit duplikaci obsahu.

Ukázku šablony lze spatřit na Obr-2.3, kde je vidět jednoduchá webová stránka složená z elementů, které na ni společně s obsahem dodává právě šablona.



Obr-2.3 – Ukázka jednoduché šablony

2.3.6 Workflow

Workflow je nástroj pro správu, plánovaní, sledování a řízení (podnikových) procesů. Proces je v tomto případě organizovaná skupina vzájemně propojených činností, které společně vytvářejí výsledky hodnotné pro zákazníky (podniku).

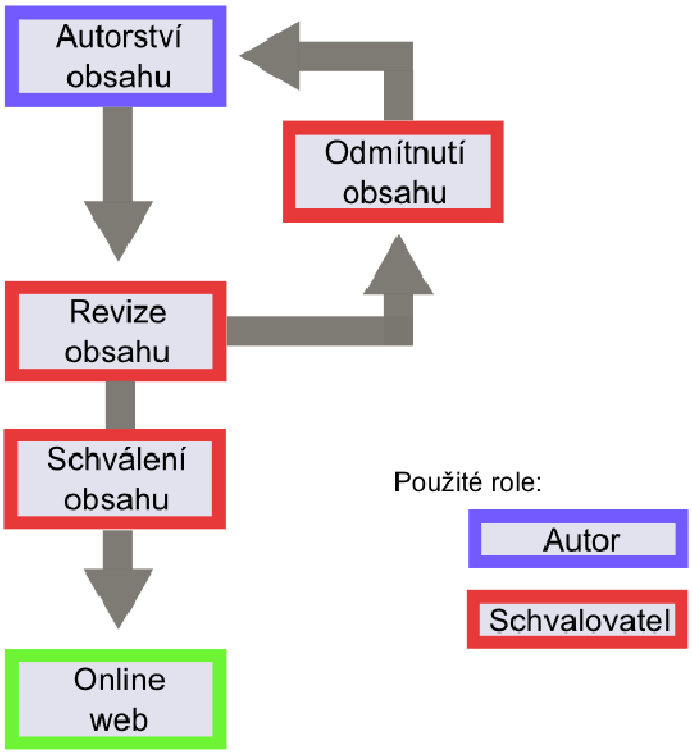
Z hlediska CMS je workflow proces vytváření cyklů sekvenčních a paralelních úloh, které musí být v CMS splněny. Pokud víme, kolik úloh musí být splněno, pak už si z toho

vypočteme, kolik lidí budeme potřebovat, kteří to udělají. To může fungovat i naopak – podle toho, kolik lidí na práci máme, si sestavíme projekt tak, abychom jim vyhověli.

Workflow je série kroků, které musí nastat v CMS a periodicky se budou opakovat. Např. následující kroky musí nastat k připravení aktualit k publikaci:

- někdo ji musí napsat, nebo získat z nějakého zdroje,
- někdo ji musí editovat,
- někdo ji musí prohlédnout a popřípadě opravit,
- někdo ji musí umístit do správné sekce,
- někdo ji musí publikovat.

Samozřejmě, že úloh může být v rámci cyklu i více i méně. Workflow by je však mělo dostatečně popisovat. Typickou ukázku workflow v CMS můžete spatřit na Obr-2.4.



Obr-2.4 – Typická ukázka workflow v CMS.

2.4 Nástroje CMS – stavební prvky

Kapitola 2.3 pojednávala o tom, jaké postupy by měl CMS obsahovat. Tato kapitola upřesní, jakým způsobem a v jaké formě mohou být v CMS implementovány.

Stavební elementy CMS zde budu popisovat v návaznosti na jednotlivé části kostry CMS, která byla představena v kapitole 2.2

2.4.1 Technologické prostředky

V této kapitole se budu soustředit na technologické prostředky užívané pro reprezentaci obsahu. Součástí této práce již nebude diskuze nad spojením CMS s databázovými systémy a programovacími technikami.

Jelikož se CMS zabývá mimo jiné strukturou a formátem obsahu, je pro jeho reprezentaci velice důležité použití nějakého značkovacího jazyka. Také proto zřejmě nenajdeme webové CMS, které by nepoužívalo značkovací jazyky HTML a XML a různé jejich standardizované odrůdy. Proč jsou tedy HTML a XML tak hojně využívané:

- reprezentují formát obsahu,
- reprezentují strukturu obsahu,
- skládají se ze značek (*tagů*), které ohraničují obsah a přidávají k němu kontext,
- dávají nám buď jednoduchou, ale omezenou sadu tagů (HTML), nebo neomezenou sadu, ale o to víc zavazující k odpovědnosti, jak budou tagy interpretovány (XML).

Dříve často platilo, že HTML bylo použito hlavně pro formátování a XML hlavně pro strukturování. HTML by se však také mělo starat pouze o strukturu. Na formátování jsou pak použity jiné techniky.

Co se týká aplikační vrstvy CMS, tak její technologické prostředky se v první řadě odvíjí od programovacího jazyka, ve kterém je implementované jádro CMS. V dnešní době však s pomocí systémové integrace mohou být technologické možnosti CMS prakticky neomezené. Některé systémy dokonce používají vlastní formátovací značky [3], nebo podporují wiki editaci. Možností je samozřejmě mnohem více, ale ty již také nebudou předmětem diskuze této práce.

2.4.1.1 XML v CMS

XML (eXtensible Markup Language) je obecný značkovací jazyk, vyvinut a standardizován konsorciem W3C. V dnešních systémech pro správu obsahu tento značkovací jazyk hraje hlavní roli. Je to vůbec jedna z nejdůležitějších věcí v CMS a jeden z nejlepších nástrojů pro správu rozkouskovaného obsahu. Umožňuje efektivní využití obsahu a hlavně

jeho znovupoužitelnost. Hlavní přínos je v oddělení od sebe formátu a obsahu, spojení obsahu s metadaty a našlo by se toho ještě více.

2.4.2 Nástroje pro systém sběru obsahu

Systém sběru obsahu je mostem mezi aplikací používanou pro inicializační vytváření obsahu a management systémem používaným pro skladování obsahu. Collection systém zahrnuje několik subsytémů, jejichž nástroje by měly dohromady postavit silný systém pro tvorbu a získávání obsahu uvnitř CMS. Tyto subsytémy popíšu v následujících podkapitolách.

Nástroji z této kapitoly by měly být pokryty požadavky kladené na přístupnost vytváření obsahu a částečně také na jeho organizaci (viz kapitola 2.1.1).

2.4.2.1 Systém pro autorství

Jeho nástroje umožňují autorům vytvářet obsah od nuly. Zde uvádím nejčastější typy obsahu, které do CMS vytváříme:

- text – může být ve formátu HTML, XML, generovaný textovým procesorem, nebo tabulkovým kalkulátorem,
- obrázky,
- video,
- zvuky.

S přihlédnutím na výše uvedené typy a možné operace potřebné pro začlenění jejich obsahu do CMS by měly být uvažovány tyto nástroje pro systém autorství:

- Integrované vyhledávání v CMS.
- Integrované workflow.
- Podpora exekuce vnějších nástrojů, kterými editujeme komponenty a elementy z CMS.
- Podpora nahrávání externích souborů (dat) do systému.
- Pokročilá podpora médií – jde např. o zacházení s obrázky jako se samostatnými komponentami a vlastní sadou metadat. Na tuto samostatnou komponentu se pak bude možné odkázat v textu.
- Spell checking – neboli kontrola pravopisu formulářů.
- Podpora vytváření obsahu offline.

- Preview vytvořeného obsahu.
- Nástroje pro získávání a prezentaci odezvy na autorský obsah.
- Podpora pro vkládání metadat.
- Další webové formuláře pro účel vytváření obsahu.

2.4.2.2 Systém konverze obsahu

Nástroje pro konverze umožňují modifikaci existujícího obsahu takovým způsobem, aby je mohl CMS využít.

Měly by být uváženy následující nástroje pro systém konverze:

- Základní import – např. obrázkovým editorem otevřu obrázek, zkonvertuji do potřebného formátu a importuji do CMS
- Nástroje pro jednoduché mapování souborů na obsahové typy CMS
- Podpora dávkového zpracování souborů – např. pro konverzi všech souborů z jednoho adresáře
- Nástroje pro monitorování kvality konvertovaných informací
- Nástroje umožňující revizi konvertovaného obsahu

2.4.2.3 Systém získávání obsahu

Nadpisem kapitoly je myšleno získávání informací z jiných CMS a vůbec externích zdrojů, tedy ne způsobem vytváření informací autory. V této sekci je k dispozici několik málo komerčních nástrojů. Může se jednat o nástroje, které pracují na bázi získávání informací z různých vnějších lokací a zabalení do standardních XML struktur. Interní systém v CMS by měl nabízet minimálně následující nástroje:

- Nástroje pro validování dat, které mají XML formát
- Nástroje pro filtrování vstupních souborů
- Transformace z jedněch XML struktur do jiných[7]
- Podpora stanovení priorit nabytých dat v závislosti na lokaci odkud přišly, zdroji a obsahovém typu

[7] např. užitím XSL

2.4.2.4 Systém shromažďování dat

Tento systém bude ty předchozí doplňovat nástroji, které se budou starat o práci kolem strukturování nabytého obsahu a o jeho metadata. Důraz je kladen tedy na přípravu obsahu pro uložení do CMS repository, a k tomu využijeme:

- Nástroj pro automatickou aplikaci metadat na obsah – alespoň ta nejdůležitější, budeme je přiřazovat na základě typu souboru, nebo lokaci. Vhodná by byla též podpora dávkové aplikace.
- Nástroje pro segmentaci obsahu – proces rozdělování obsahu do komponent a elementů, se kterými potřebuje CMS pracovat
    - Nástroje pro oddělení elementů od okolního obsahu
    - podpora vyhledávání elementů
    - podpora indexace elementů
- Nástroje pro zpětnou vazbu od editorů k autorům – např. možnost editorů komentovat obsah vytvořený autory
- Podpora různých jazykových variant pro obsah
- Nástroje pro podporu znovupoužitelnosti komponent. Jako příklad uvedu možnost zobrazení preview komponenty autora na všech místech, kde může být komponenta zobrazena
- Nástroje a techniky pro aplikaci metadat na obsahové komponenty, pro co nejjednodušší práci s metadaty. Mohou se např. vyskytovat i nástroje pro podporu importu a exportu metadat z databáze, nebo externích souborů. Běžné jsou i nástroje využívající metadata pro vytváření různých uspořádaných seznamů elementů a komponent.

2.4.3 Nástroje pro systém správy obsahu

Nástroje správy obsahu by měly být spojeny s uživatelským rozhraním přímo napojeným na správu centrálního úložiště, které bývá nejdůležitějším prvkem CMS.
S nadsázkou se dá říci, že by v CMS mělo zastávat roli „operačního systému“. Všechny další nástroje se pak odvíjí právě od možností tohoto úložiště.

Nástroji z této kapitoly by měly být pokryty požadavky kladené na organizaci a správu obsahu a na snížení nákladů na jeho údržbu (viz kapitola 2.1.1).

2.4.3.1 Repository

Základním nástrojem úložiště je prostor pro skladování obsahu. K tomu může být využito relační, či objektové databáze. Pro operace s obsahem v tomto úložišti by měly být dále uváženy nástroje pro ukládání obsahových komponent, obsahových typů a vztahů mezi nimi. Nástroje pak budou také závislé na tom, zda jsou data ukládána do tabulek v relační databázi, nebo do XML souborů.

Některé systémy mohou mít také implementovaný vlastní virtuální souborový systém, který může být uložen kompletně v relační databázi a s reálným souborovým systémem bude komunikovat přes systém synchronizace obsahu.

Repository by mělo dále poskytovat nástroje pro:

- Verzování obsahu - CMS může podporovat automatické, nebo manuální verzování obsahu popř. kombinaci obojího. Verzovací nástroje by měli podporovat alespoň vlastnosti, jako obnovení předchozí verze obsahu, porovnávání rozdílů mezi vybranými verzemi, větvení verzí, sloučení změn z různých větví apod.
- Lokalizaci obsahu – Podpora různých jazykových variant obsahových komponent i obsahových elementů. Lokalizace obsahu by pak měla být brána v potaz i ve vyhledávacím stroji a v šablonách, které vytváří publikace.
- Sdílení obsahu – Podpora zamykání obsahu, oznámení o změně obsahu atd.

2.4.3.2 Administrace CMS

Administrace pokrývá správu vlastností a rysů CMS, které nemají přímou spojitost s obsahovými komponentami. Měla by pokrývat globálně správu celého systému a k tomu poskytovat nástroje např. pro:

- Logování
- Zálohování obsahu
- Správu databázových rozhraní
- Nastavení přístupových práv k obsahu a s tím správu uživatelů v CMS

2.4.4 Nástroje pro systém publikování obsahu

Primárním úkolem těchto nástrojů je převést obsah z CMS repository na publikace, které je možné zveřejnit na webu. V systému publikování hraje nejdůležitější roli šablonový systém, který popíši v kapitole 2.4.4.1. Dále tato sekce zahrnuje navíc nástroje pro vytváření

tiskových sestav, odesílání obsahu ve formě emailových zpráv, transformaci obsahu na webové syndikace a systém personalizace. U posledně jmenovaného můžeme očekávat nástroje pro tvorbu uživatelského rozhraní, které umožní jednoduchou personalizaci stránek dle koncového uživatele. Důležitou roli v tomto hraje nastavení určitých pravidel personalizace, dále nástroje pro vytváření profilů koncových uživatelů a jejich uchování.

Nástroji z této kapitoly by měly být pokryty požadavky kladené na zefektivnění procesů publikace obsahu (viz kapitola 2.1.1).

2.4.4.1 Šablonový systém

Hlavními nástroji využitelnými ve spojitosti se systémem publikování budou zajisté nástroje pro vytváření šablon, které mají za úkol sestavit publikace z obsahových elementů. Šablonový systém by měl pak umožnit vytvářet jak statické, tak dynamické publikace.

Důležitým prvkem šablon jsou elementy navigace. CMS by měl tedy poskytovat přehledný systém pro tvorbu navigačních prvků, který umožní uchovávat navigační prvky hierarchicky, vytvářet indexy mezi nimi a vytvářet jejich sekvence.

Šablonový systém by také měl poskytnout nástroje pro jednoduché sestavení layoutu publikací.

Tímto končí kapitola věnovaná obecnému popisu systémů pro správu obsahu. V té následující se budu věnovat implementačním detailům jednoho konkrétního CMS, který se nazývá OpenCms.

3 Implementační detaily OpenCms

OpenCms [11] je web content management system na profesionální úrovni vyvíjený německou společností Alkacon Software GmbH [14]. Jedná se o open-source software dostupný pod licencí LGPL, která osvobozuje software od licenčních poplatků a umožňuje jeho bezplatné užívání. První open-source verze systému byla uvolněna v březnu roku 2000 a momentálně (červen 2009) aktuální stabilní verze nese označení OpenCms 7.0.5.

OpenCms je systém založený na Java a XML technologiích, vhodný především pro tvorbu podnikových webových aplikací a intranetu. Je stavěný pro potřeby středních až větších podniků s již existující IT infrastrukturou. Předností systému je jeho flexibilita a uživatelská přizpůsobitelnost.

K nasazení OpenCms je potřeba mít nainstalovaný podporovaný databázový systém, Java development kit minimálně verzi 1.4 a webový Servlet/JSP kontejner. Výhodou je kombinace webového kontejneru s webovým serverem. Podporovaný podmíněný software ukazuje Tab-3.1. Jako systém postavený na Java platformě je pak OpenCms nezávislý na operačním systému.

Typ software	Podporované aplikace
Webový kontejner	Tomcat, Jboss, Resin 3, Websphere 6, BEA Weblogic
Databáze	Oracle, MySQL, PostgreSQL, MS SQL Server, DB2, AS400, HSQL
Webový server	IIS, Apache

Tab-3.1 – Software podporovaný OpenCms, který vychází ze specifikace na [15]

V této kapitole dále popíši architekturu OpenCms a hlavní charakteristické prvky, které nabízí pro správu obsahu. Ve větším detailu pak představím klíčové prvky systému a nakonec nastíním, jakou nabízí podporu pro proces vývoje nových modulů a funkčností.

3.1 Charakteristické rysy OpenCms

Jelikož OpenCms je projekt, který se již nějakou dobu vyvíjí, seznam nástrojů a vlastností poskytovaných pro správu obsahu je již poměrně velký a proto zde nebudu vypisovat všechny, ale pouze ty nejdůležitější. Kompletní seznam vlastností systému lze najít na [11].

- **Mechanismus projektů –** Slouží k rozdělení repository na související části. V systému je možné vytvářet projekty dle libosti. Nejdůležitější jsou však dva základní projekty, které se nachází v každé instalaci OpenCms a s kterými si většinou vystačíme.
  - Online projekt – Obsahuje data, která jsou viditelná na frontend stránkách všem uživatelům, kteří k tomu mají práva. Obsah v tomto projektu nelze editovat.
  - Offline projekt je aktuální pracovní verze obsahu, který je viditelný pouze z Workplace explorer a pouze pro uživatele, kteří mají do exploreru přístup. V Offline projektu je možné obsah libovolně editovat, vytvářet a mazat. Offline projekt je téměř identickou kopií Online projektu s tím rozdílem, že obsahuje změny provedené od poslední publikace. Publikace je pak akce, která převede změny z Offline projektu na Online projekt.
- **Mechanismus modulů –** Umožňuje vhodně zaobalovat obsah a funkčnost pro pohodlné nasazení na jiné instalaci OpenCms. Detailní popis zahrnuje kapitola 3.5.
- **Systém přístupových práv k obsahu –** Umožňuje volitelně omezit přístup k veškerému obsahu OpenCms. Detailní popis zahrnuje kapitola 3.4.3.
- **Šablonový systém –** Umožňuje vytvářet vysoce dynamické webové stránky. Detailní popis zahrnuje kapitola 3.6.1.1.
- **Integrovaný WYSIWYG editor pro nestrukturovaný obsah –** Umožňuje editaci stránek s nestrukturovaným obsahem. Poskytuje podporu pro snadné vytvoření layoutu stránky, snadné vkládání obrázků, odkazů, vytváření tabulek apod.
- **XmlContent –** Funkcionalita podporující snadnou definici strukturovaného obsahu založeného na XML. OpenCms také umožňuje automaticky vygenerovat editor pro tento typ obsahu.
- **Integrovaný full-textový vyhledávací engine –** OpenCms nabízí full-textové vyhledávání prostřednictvím open-source vyhledávacího enginu *Apache Lucene* [16].

Ten podporuje v OpenCms vyhledávání ve vybraných typech obsahu, umožňuje různé způsoby indexování obsahu pro vyhledávání apod. V OpenCms 7.0.0 navíc přibyla možnost vyhledávání i ve Workplace a nejen na frontend stránkách.

- **Sada konfigurovatelných meta informací (property položek) pro všechny resource objekty v repository –** Workplace poskytuje různé dialogy pro nastavení property položek na resource objektech. Meta informace se mohou dědit v podstromu adresářové struktury. OpenCms také poskytuje rozhraní, které umožňuje property položky snadno využívat v aplikační vrstvě i zobrazovat v prezentační vrstvě.

Velkou výhodou je také možnost integrovat do systému frameworky a knihovny třetích stran. Příkladem ověřených integrovaných prvků jsou:

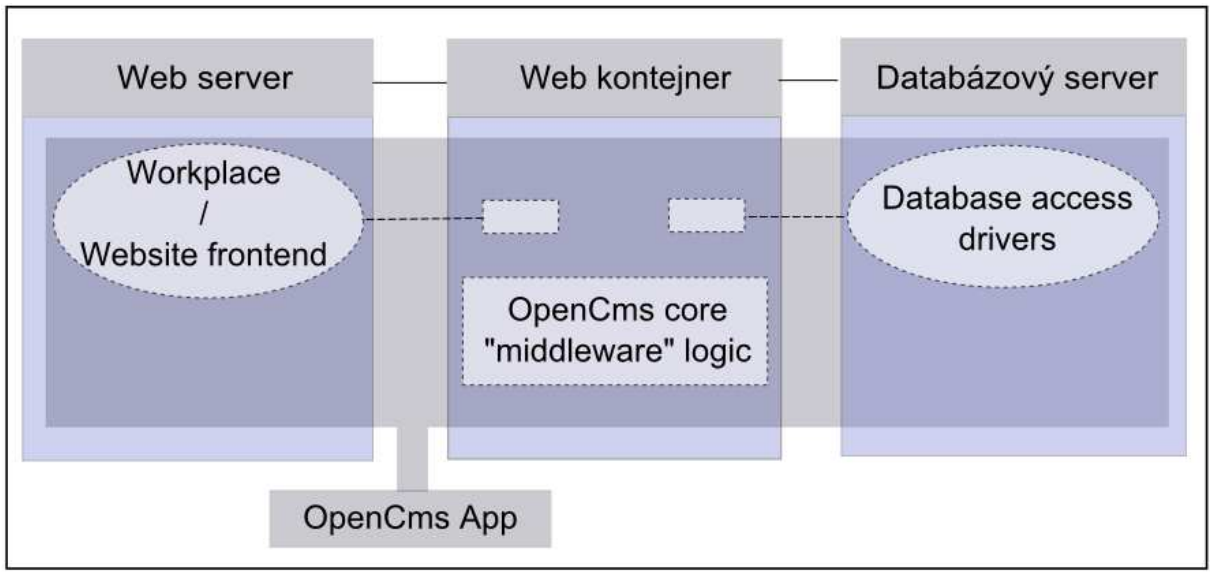
- Apache Struts,
- Spring Framework,
- Java Server Faces (JSF),
- Groovy.

Ve standardní instalaci (mluvím o verzi 7.0.0) pak OpenCms poskytuje některé integrované open-source komponenty, které se staly standardní výbavou. Jedná se např. o:

- Apache Lucene – vyhledávací engine,
- DOM4J – XML API,
- JTidy – HTML parser,
- Log4j – Logovací API,
- jQuery – JavaScript knihovna.

3.2 Architektura OpenCms

OpenCms se svou strukturou přizpůsobuje typické J2EE aplikaci s třívrstvou aplikační architekturou, kterou ilustruje Obr-3.1. Tak popisuje základní OpenCms návrh kniha [9].



Obr-3.1 – Třívrstvá architektura OpenCms

OpenCms aplikace má sama o sobě také tři vrstvy:

- Prezentační vrstvu – Workplace / webové stránky na frontendu.
- Aplikační vrstvu – OpenCms aplikační jádro, jehož součástí je rozsáhlé API poskytující podporu pro vývoj modulů a funkčností v OpenCms.
- Datovou vrstvu – Databázové drivery.

Obr-3.1 naznačuje, že tyto vrstvy běží kompletně na webovém kontejneru. Požadavky na webový server jsou vytvářeny v aplikaci zvané Workplace a na frontend webových stránkách. Webový server tyto požadavky zpracovává a dále posílá webovému kontejneru, který je předá na Servlet z jádra OpenCms. Ke čtení a zapisování obsahu pak OpenCms využívá databázový server, s nímž je spojen pomocí svých jádrových tříd zvaných „drivers“. Napojení datové vrstvy na databázový server se v OpenCms nastavuje při instalaci systému. Libovolně lze do OpenCms přidávat další uživatelské datové vrstvy, které spojíme s databázovým serverem přes konfigurační soubory systému.

3.2.1 Souborové systémy v OpenCms

OpenCms, jako každý systém pro správu obsahu, disponuje centrálním úložištěm obsahu (repository), které podporuje celou řadu funkcí pro správu obsahu. Repository je kompletně uložené ve vlastním Virtuálním souborovém systému (VFS), kterému se říká virtuální z toho důvodu, že je umístěn v databázi a ne na pevném disku.

Naproti tomu nesmíme zapomenout, že OpenCms musí být zároveň uložen jako webová aplikace i v reálném souborovém systému (RFS).

3.2.1.1 OpenCms v RFS

Na reálném souborovém systému operačního systému reprezentuje OpenCms pouze adresář webové aplikace, umístěný v kontextu s použitým webovým kontejnerem. Struktura tohoto adresáře by pak měla být povědomá všem vývojářům webových aplikací. Tento adresář budu označovat *OPENCMS_RFS_FOLDER*.

Dále nebudu popisovat kompletní strukturu webového adresáře, zmíním pouze důležité části a ty se nachází v podadresáři *WEB-INF*. Struktura této složky je:

- **classes –** Složka zahrnuje Java class soubory a Java properties soubory, které nejsou součástí žádného Java balíku.
- **config –** Složka zahrnuje konfigurační soubory OpenCms, jejichž popis najdete v Tab-3.2. Některá nastavení z těchto souborů je možné měnit i přes Administrační část OpenCms workplace (viz kapitola 3.3.2).
- **jsp –** Složka zahrnuje JSP soubory vyexportované z VFS. JSP se do tohoto místa exportují, kvůli tomu, aby byly přístupné webovému kontejneru a mohli být zkompilovány.
- **lib –** Složka obsahuje JAR knihovny s OpenCms API a knihovny externích open-source komponent, které OpenCms využívá. Zároveň se do této složky budou exportovat JAR knihovny, které jsou součástí různých modulů, pokud budou v modulu označeny jako komponenty k exportu do RFS. Základní knihovna obsahující OpenCms API se nazývá *opencms.jar*.
- **packages –** Složka zahrnuje soubory vyexportovaných balíků z VFS za účelem importu do jiné instalace systému. Jedná se o balíky s OpenCms moduly, nebo s vyexportovaným obsahem databáze.

Konfigurační soubor	Popis
opencms.propeties	Zahrnuje důležitou konfiguraci potřebnou pro start systému. Nejdůležitější částí souboru je nastavení připojení k databázi.
opencms.xml	Hlavní konfigurační soubor zahrnující názvy tříd použitých ke konfiguraci systému. Každá z těchto tříd implementuje rozhraní I_CmsXmlConfiguration a je inicializována při startu systému.
opencms-system.xml	Soubor pro nastavení jádra systému. Zahrnuje nastavení lokalizace, cache, default uživatelů, definice *Site*. Většina nastavení v tomto souboru musí být změněna manuálně právě zde.
opencms-vfs.xml	Zahrnuje nastavení týkající se VFS. Najdeme zde např. registrované resource objekty a jejich zaváděcí třídy, nastavení widget prvků. Je možné zde také registrovat nový typ resource objektu.
opencms-workplace.xml	Zahrnuje nastavení týkající se Workplace exploreru. Vyžaduje také editaci, pokud chceme do systému registrovat nový typ resource objektu.
opencms-search.xml	Zahrnuje nastavení související s vyhledávacím enginem a indexy vyhledávání. Nastavitelné je i vyhledávání v metadatech obsahu. Většinu nastavení tohoto souboru lze měnit v Administrační části OpenCms workplace.
opencms-importexport.xml	Zahrnuje nastavení importu a exportu souborů z VFS.
opencms-modules.xml	Zahrnuje seznam registrovaných modulů v OpenCms i s jejich nastavením. Většina nastavení tohoto souboru lze měnit v Administrační části workplace. Pokud však chceme k modulu přidat nový typ resource objektů, provedeme editaci tohoto souboru.

Tab-3.2 – Konfigurační soubory OpenCms.

3.2.1.2 OpenCms ve VFS

Jak bylo již řečeno, VFS je kompletně uloženo v databázi. Důvodem je nezávislost na použitém operačním systému a to přináší další výhody. VFS je velice podobný reálnému souborovému systému, který podporuje adresáře, soubory a přístupová práva k nim. Díky nezávislosti na RFS však může vytvářet vlastní typy adresářů i souborů a může k nim poskytovat přístupová práva na základě vlastního systému. Dále nabízí VFS řadu další funkčnosti a operací, které jsou typické pro CMS.

Najdou se však také nevýhody VFS. Mohl bych zmínit např. problémy, které mohou nastat, pokud do VFS nahráváme obsah z RFS, jehož typ není ve VFS definován. V tom

případě se do VFS uloží jako binární soubor, bez ohledu na to, zda se jedná např. o obrázek. Tento problém se dá vyřešit nastavením mapování souborů RFS na soubory VFS v konfiguračním souboru *opencms-vfs.xml*. Další problém vznikne při vývoji např. JSP souborů pomocí vhodného IDE. To samozřejmě standardně nemá přístup do VFS, avšak opět se to dá obejít pomocí nástroje pro synchronizaci VFS a RFS dostupného v OpenCms repository popř. nastavením přístupu WebDAV do VFS, který OpenCms poskytuje od verze 7.0.0.

3.3 Uživatelské rozhraní OpenCms – Workplace

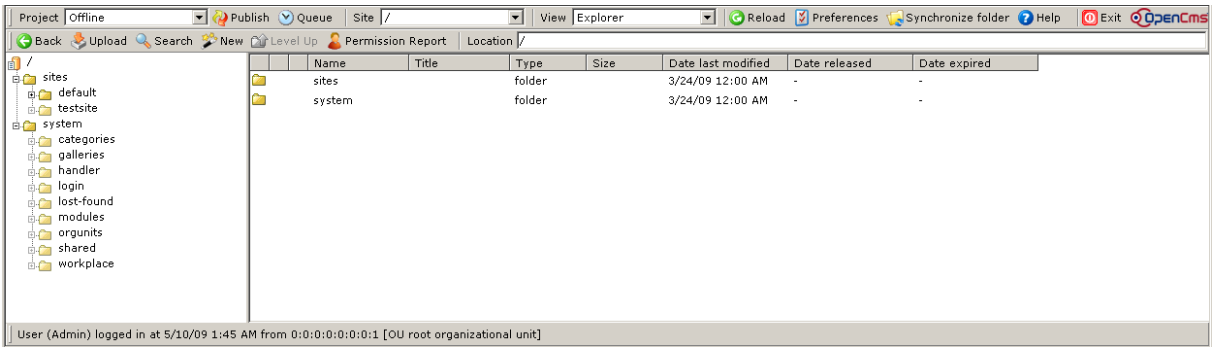
Workplace je webové rozhraní, pro správu OpenCms repository, které se skládá ze dvou částí – ze souborového průzkumníka, který se nazývá „Explorer view“ a z administrační části systému, ve které najdeme nástroje pro správu celého systému OpenCms. Druhá část nese název „Administration view“.

3.3.1 Explorer view

Workplace explorer, neboli průzkumník VFS je výchozím pohledem Workplace a umožňuje procházet celý strom virtuálního souborového systému. Již na první pohled je patrná jeho podoba s Průzkumníkem z operačního systému Microsoft Windows, podobná je i funkcionalita.

Explorer viditelný na Obr-3.2 se skládá ze tří hlavních částí. Horní část pokrývá tlačítková lišta poskytující rychlý přístup k nejčastějším systémovým akcím. V levé hlavní části se nachází strom adresářové struktury repository a v pravé hlavní části pak seznam resource objektů z aktuálně zvolené složky ve VFS stromu.

Workplace explorer je velká JavaScript aplikace, jejíž nevýhoda je poměrně nesnadná přizpůsobitelnost. Pouhé přidání tlačítka do lišty vyžaduje zásah do jádra OpenCms, což není v tomto případě zrovna triviální operace. Více o tomto tématu zmíním v kapitole 5.3.3.3.



Obr-3.2 – Explorer view

3.3.2 Administration view

Webové rozhraní určené pro správu OpenCms. Najdeme zde řadu nastavení základních funkcí a nástrojů OpenCms, z nichž většina je přístupná pouze uživatelům ze skupiny *Administrators*, nebo na základě konkrétní role přiřazené uživateli. Administrační pohled ilustruje Obr-3.3.

Obr-3.3 – Administration view

Všechna nastavení dostupná přes Administration view se dají zároveň měnit v RFS v konfiguračních souborech. Naopak to však neplatí – ne všechna nastavení z konfiguračních souborů se dají měnit prostřednictvím Administration view. Doporučení pro správu systému je takové, že pokud je požadované nastavení dostupné v Administration view, mělo by být modifikováno právě tam.

OpenCms umožňuje jednoduše přidávat vlastní položky do Administration view (viz poslední položka na obr-3.3). Podrobný popis této možnosti blíže popisuji v kapitole 5.3.3.3.

3.4 Repository

Repository uložené ve VFS je místo, kam je v CMS ukládán obsah včetně jeho metadat. OpenCms repository poskytuje řadu nástrojů pro správu tohoto obsahu. Nástrojem, skrze který je pak možné repository spravovat a zmiňované nástroje používat, je aplikace nazvaná Workplace viz kapitola 3.3.

3.4.1 Adresářová struktura VFS

VFS strom se skládá ze dvou hlavních podstromů s adresářovou strukturou. K tomu, aby mohl procházet oba tyto podstromy, a tedy kompletní VFS strom, musí být uživatel členem skupiny *Administrators*, nebo musí mít přiřazenou roli *VFS resource manager*. Navíc musí mít nastaven kořenový *Site* adresář jako '/', který standardně není vybrán při startu systému (To se však dá změnit v dialogu pod tlačítkem *Preferences* a záložkou *Workplace)*.

Pokud tedy máme zvolen kořenový adresář, pak uvidíme VFS strom skládající se z následujících dvou částí:

- *sites* – Každý adresář nacházející se pod touto větví stromu reprezentuje jeden kompletní web, který se nazývá *site*. Tímto jsou od sebe logicky odděleny resource objekty pro různé *Site* položky. Pokud jste si všimli na Obr-3.2 vysunovací nabídky s označením „Site", tak věžte, že v této nabídce si lze zvolit pro procházení v podstromu právě pouze jednu ze *Site* položek z tohoto adresáře[1]. Přidání nové *Site* do systému pak obnáší nejen vytvoření nového adresáře přímo v adresáři *sites*. Navíc je třeba přidat záznam nové *Site* do konfiguračního souboru *opencms-system.xml*.
- *system* – Tato větev představuje systémový adresář VFS, zahrnující vesměs implementaci jádra OpenCms a samotného Workplace. Dále se zde nachází složka *modules*, která obsahuje všechny moduly OpenCms, složka *handler* se šablonami pro chybové stavové kódy HTTP (404, 500 atd.), složka *workplace* obsahující resource objekty implementující celý Workplace (Explorer i Administrační část) a další.

Při vývoji nových modulů v systému je třeba dát pozor na rozlišení dvou různých URI, se kterými pracuje OpenCms. Jedná se o kořenovou URI systému a relativní URI vzhledem

[1] Ve vysunovací nabídce je *Site* reprezentována property položkou *Title* nastavenou na adresář *Site*.

k vybrané *Site* položce. Tento problém je blíže popsán v závěru implementace modulu „Report s přístupovými právy“ v kapitole 5.3.4.

3.4.2 Jednotky VFS stromu – resource objekty

Obsah je VFS reprezentován a tříděn pomocí souborů a adresářů, které se v OpenCms nazývají společně resource objekty. Jsou totiž implementovány pomocí třídy *org.opencms.file.CmsResource*, která je velice důležitým objektem pro používání OpenCms API. Pro rozlišení adresářů a souborů pak existují dva typy odvozené od této třídy:

- *org.opencms.file.CmsFile* – reprezentuje soubor ve VFS,
- *org.opencms.file.CmsFolder* – reprezentuje adresáře ve VFS.

Pokud mám k dispozici resource objekt, pak mohu využít metody `isFile()` a `isFolder()` k zjištění konkrétní podtřídy. Přesto většinou při práci s resource objekty budete využívat spíše třídu *CmsResource*,která obsahuje operace pro manipulaci se soubory i adresáři a která se od *CmsFile* liší vlastně pouze tím, že *CmsFile* poskytuje navíc přístup k samotnému obsahu souboru. Ten většinou v Java třídách nebudeme potřebovat. To platí i pro *CmsFolder*, která je téměř identická s *CmsResource* a její instanci využijeme pouze ve speciálních případech. Zde se nabízí pouze využití metody `isFolder()` pro kontrolu, zda je resource objekt adresář a tudíž může obsah další resource objekty jako potomky.

OpenCms API umožňuje nad resource objekty provádět operace jako zobrazení historie změn resource objektu, obnova již smazaných objektů, publikace pouze vybraných objektů, vrácení poslední nepublikované změny na objektu, uzamknutí objektu, nastavení přístupových práv k objektu, editace meta informací objektu a další operace známé z RFS operačního systému.

3.4.2.1 Typy resource objektů v OpenCms

Každý resource objekt v OpenCms patří k nějakému inicializovanému typu v systému. Všechny dostupné typy resource objektů pak obsahuje konfigurační soubor *opencms-vfs.xml*. Tyto typy jsou inicializovány při startu systému. Současně se v tomto konfiguračním souboru nastavuje mapování souborů přicházejících z RFS na typy resource objektů ve VFS.

Samotná třída *CmsResource* obsahuje informaci o typu pouze v podobě *Integer* parametru, který představuje unikátní identifikátor deskriptoru příslušného typu. K těmto

deskriptorům se lze pak dostat přes třídu *org.opencms.loader.CmsResourceManager,* která shromažďuje všechny zavaděče, typy a kolektory resource objektů. K této třídě se za běhu systému dostaneme staticky pomocí třídy *org.opencms.main.OpenCms,* poskytující statické metody použitelné jinými třídami k přístupu k základním systémovým operacím a vlastnostem.

Deskriptory typů resource objektů v OpenCms definuje rozhraní *org.opencms.file. types.I_CmsResourceType.* Rozhraní určuje rámec pro konkrétní třídy typů resource objektů, které si pak mohou implementovat určité operace specifické právě pro konkrétní typ. Pokud bychom chtěli přidat nový typ resource objektu do systému, je vhodné třídu typu oddědit od abstraktní třídy *org.opencms.file.types.A_CmsResourceType*, implementující předešlé rozhraní.

Deskriptory standardních typů resource objektů (viz Tab-3.3) definovaných v OpenCms jsou umístěny v balíku *org.opencms.file.types.*

Typ resource objektu	Třída deskriptoru
binary	CmsResourceTypeBinary
folder	CmsResourceTypeFolder
extended folder	CmsResourceTypeFolderExtended
image	CmsResourceTypeImage
jsp	CmsResourceTypeJsp
plain	CmsResourceTypePlain
pointer	CmsResourceTypePointer
unknown	CmsResourceTypeUnknown
unknown_file	CmsResourceTypeUnknownFile
unknown_folder	CmsResourceTypeUnknownFolder
xmlcontent	CmsResourceTypeXmlContent
xmlpage	CmsResourceTypeXmlPage

Tab-3.3 – Typy resource objektů v OpenCms

XmlContent

Zajímavým typem resource objektu je *XmlContent*. Ten představuje typ se strukturovaným obsahem a lze pomocí něj snadno vytvářet obsahové typy, jako např. aktuality, události, články, formuláře, ale také různé konfigurační typy resource objektů využívaných uvnitř Workplace. *XmlContent* je součást datové vrstvy, protože určuje pouze strukturu informací, které nese. Pokud mají být informace i prezentovány (nejedná-li se např.

o konfigurační soubor), pak prezentační vrstvu tvoří šablony, které *XmlContent* zpracovávají. *XmlContent* poskytuje následující výhody:

- OpenCms dokáže pro *XmlContent* typ automaticky vygenerovat editační formulář ve Workplace, který editaci velice usnadňuje.
- OpenCms poskytuje funkce pro snadné procházení a prezentování obsahu v *XmlContent* souborech. Funkce jsou dostupné v OpenCms JSP taglib knihovně.
- *XmlContent* soubory jsou uložené ve VFS a podporují všechny operace dostupné pro *CmsResource* objekty.
- *XmlContent* soubory podporují vnořování, takže je lze používat i v jiných *XmlContent* souborech a vytvářet tak pokročilé obsahové struktury.

Pro vytvoření nového typu *XmlContent* resource objektu a jeho editoru stačí provést následující kroky:

1. Vytvořit XSD (XML schema definition) soubor – Definuje strukturu, formát a datový typ *XmlContent* souboru.
    a. XSD musí zahrnovat definici jednotlivých elementů obsahu, jejich datové typy a widget prvky. Widget prvky jsou formulářové prvky, které budou vygenerovány v editačním formuláři za účelem zadání hodnoty pro příslušný obsahový element. OpenCms lze rozšiřovat o nové widget prvky v nových třídách, které zdědíme od abstraktní třídy, která je poskytnuta právě pro tvorbu nových widget prvků. Jedná se o třídu *org.opencms.widgets.A_CmsWidget.*
2. Registrace *XmlContent* typu v rámci některého modulu OpenCms zajistí, že bude možné vytvářet ve VFS nové obsahové typy příslušného *XmlContent.*
3. Vytvoření šablon pro zobrazení *XmlContent* typu. Tato akce není součástí vytvoření *XmlContent* typu, ale zajistí prezentaci obsahu na webu. V šablonách se ke *XmlContent* typu přistupuje pomocí tagů přímo určených ke zpracování *XmlContent* z OpenCms JSP taglib knihovny.

3.4.3 Systém přístupových práv

OpenCms disponuje velice dobře zpracovaným, vyzrálým bezpečnostním modelem, postaveným na objektech uživatelů, skupinách uživatelů, rolích uživatelů a jejich kontrolovaném přístupu k resource objektům ve VFS. Tento přístup je volitelně omezen

pomocí propracovaného systému přístupových práv, který uživatelům nastavuje různá oprávnění ke konkrétnímu resource objektu.

3.4.3.1 Uživatelé, skupiny a role

Přístupová práva se definují pro tři typy objektů, se kterými může být spojen uživatel systému OpenCms (viz Tab-3.4 – podkladem informací z tabulky je [12]).

Objekt	Reprezentativní třída	Stručný popis
uživatel (User)	org.opencms.file.CmsUser	Reprezentuje konkrétní osobu v systému přístupových práv. Je identifikován unikátním jménem. Může být zařazen do libovolného počtu skupin a mít nastaven libovolný počet rolí.
skupina (Group)	org.opencms.file.CmsGroup	Slučuje libovolný počet uživatelů. Skupině lze nastavit rodičovskou skupinu, od které zdědí přístupová práva. Nemůže mít přiřazenou roli.
role (Role)	org.opencms.security.CmsRole	Role se v OpenCms využívají pro kontrolu, zda má uživatel přístup k systémovým funkcím, které nemusí být vždy spojeny s resource objektem ve VFS. Např. může jít o kontrolu, zda má uživatel přístup k naplánování úkolu pro jiného uživatele.

Tab-3.4 – Uživatelské objekty přístupových práv.

Uživatel a skupina mají v OpenCms společného předka. Je jím „principal objekt“, který reprezentuje třída *org.opencms.security.CmsPrincipal*, obsahující běžné operace společné pro uživatele i skupiny a navíc nějaké pomocné funkce pro zacházení s instancemi principal objektů.

OpenCms umožňuje vytvářet, mazat a spravovat nové uživatele i skupiny. Jinak je tomu však u rolí. Role jsou ve skutečnosti pouze speciálním druhem skupin. Odlišují se od nich tím, že se jedná o v systému předdefinované skupiny, které nelze měnit, přidávat nové, ani je mazat. Jedině lze přiřazovat role uživatelům. OpenCms používá následující definované role:

Role	Rodičovská role	Popis
Root Administrator	None	Uživatelé s touto rolí mají přístup do celého systému.
Workplace manager	Root Administrator	Uživatelé s touto rolí mohou přistupovat k určitým funkcím ve Workplace.
Database manager	Root Administrator	Tato role umožňuje uživatelům vytvářet, mazat importovat, exportovat a editovat moduly.
Administrator	Root Administrator	Uživatelé s touto rolí mají přístup k celé organizační jednotce, ke které jsou přiřazeni.
Project manager	Administrator	Tato role umožňuje uživatelům spravovat projekty.
Account manager	Administrator	Tato role umožňuje uživatelům spravovat uživatele a skupiny.
VFS manager	Administrator	Tato role umožňuje uživateli spravovat jakýkoli resource objekt uvnitř přiřazené organizační jednotky.
Developer	VFS Manager	Tato role umožňuje uživatelům spravovat JSP soubory.
Database manager	Administrator	Uživatelé s touto rolí se mohou přihlásit do workplace.

Tab-3.5 – Role definované v OpenCms (tabulka převzatá a upravená podle [9]).

V Tab-3.5 si lze všimnout, že role mohou mít svou rodičovskou roli. Ty jsou totiž v OpenCms uspořádané hierarchicky, takže přidáním členství některé z rodičovských rolí se přidá uživateli navíc členství všech potomků tohoto rodiče.

Poněkud komplikovanější pohled na uživatelské typy může tvořit fakt, že ve standardní instalaci OpenCms jsou dostupné některé defaultní skupiny (viz Tab-3.6), které implicitně obsahují role. Je to z důvodu zachování kompatibility s předchozími verzemi OpenCms. Odstranění jedné z těchto skupin je sice možné, ale po této akci se již nepodaří znovu spustit OpenCms systém.

Skupina	Implicitní role
Administrators	Root Administrator
Guests	Žádná role
Projectmanagers	Workplace user, Project manager
Users	Workplace user

Tab-3.6 – Defaultní skupiny a jejich role

3.4.3.2 Položky řízení přístupu

Na začátku kapitoly o přístupových právech bylo řečeno, že jejich systém v OpenCms definuje oprávnění uživatelů ke konkrétnímu resource objektu. Ve skutečnosti to vypadá tak, že soubor těchto oprávnění, jeden principal objekt a jeden resource objekt tvoří jednu položku řízení přístup (access control entry), která je reprezentována třídou *org.opencms.security. CmsAccessControlEntry* (dále ACE). Jeden resource objekt pak může obsahovat několik takových položek, které má uspořádané v seznamu, který reprezentuje třída *org.opencms. security.CmsAccessControlList* (dále ACL).

V OpenCms je u každé ACE definován soubor oprávnění, které ukazuje Tab-3.7. Každé z těchto oprávnění může nabývat netradičně jednoho ze tří stupňů přístupu:

- **Allowed –** Přístup povolen.
- **No permission –** Zaškrtlé není ani *allowed,* ani *deny.* Jedná se o defaultně zmítnuté oprávnění tzv. „soft deny".
- **Deny –** Přístup zamítnut. Jedná se o tzv. „hard deny", který pokud je nastaven na oprávnění resource objektu, tak přepíše i povolení stejného oprávnění ve svém VFS podstromu.

Oprávnění	Zkratka	Popis
Read	r	Oprávnění číst obsah resource objektu.
View	v	Oprávnění určující viditelnost resource objektu ve workplace explorer
Write	w	Oprávnění zapisovat obsah do resource objektu.
Control	c	Oprávnění k nastavení přístupových práv na resource objektu
Direct publish	d	Oprávnění přímo publikovat resource objekt i bez potřebného oprávnění k publikaci celého projektu.

Tab-3.7 – Soubor oprávnění dostupných v OpenCms

Položky ACE je možné volitelně dědit na podsložky a soubory resource objektů ve VFS. Současně je možné na resource objektu nastavit, aby se zděděné ACE položky přepisovaly těmi přímo nastavenými (nelze však přepsat „hard deny").

Závěrem bych rád shrnul informace z této kapitoly. Přístup uživatele k resource objektu je výsledkem funkce, ve které se traversují ACE položky zahrnující uživatele, nebo jeho skupiny, nebo jeho role ve dvojici s oprávněními definovanými přímo na resource objektu, nebo se zděděnými. Přístup skupiny je pak výsledkem funkce, ve které se traversují

ACE položky zahrnující skupinu, nebo jejího rodiče (popř. jejich implicitní role, pokud se jedná o jednu z defaultních skupin OpenCms) ve dvojici s oprávněními definovanými přímo na resource objektu, nebo se zděděnými.

3.5 Moduly v OpenCms

Moduly jsou klíčovým prvkem OpenCms. Poskytují možnost svazovat dohromady spolu související komponenty do jednoho balíku, který je pak snadné instalovat do jiného OpenCms systému. Vůbec celé repository v OpenCms má modulární architekturu, takže dokonce i jednotlivé systémové části jsou distribuovány jako moduly. Díky tomu je celý systém snadno rozšiřitelný a aktualizace jednoduchá.

Součástí modulu mohou být šablony, obsahové typy, Java třídy a JAR balíky s aplikační vrstvou, JSP stránky a další doplňky jako např. CSS styly, JavaScript soubory apod.

3.5.1 Modul ve VFS

Moduly ve VFS jsou reprezentovány sadou adresářů a souborů, jejichž základnu tvoří adresář umístěný pod */system/modules/*. Vytvářejí a spravují se v administrační části VFS a obsahují následující povinné parametry:

- **Package name –** Určuje unikátní jméno modulu a zároveň jméno adresáře modulu ve VFS. Pojmenování se musí řídit konvencemi Javy pro pojmenování package. Package name lze zadat pouze při vytváření modulu. Později již přes administrační část nemůže být změněno.
- **Module version –** Číslo verze modulu rozdělené znakem ‘.’ na maximálně čtyři úrovně. Čísla verze lze manuálně měnit v administrační části při vydání nové verze modulu. Navíc poslední část čísla se automaticky inkrementuje o hodnotu ‘1‘ při exportu modulu do RFS.

Další nepovinné parametry modulu jsou: Popisné jméno modulu (pojmenování nevyžadující dodržení konvencí), krátký popis vlastností modulu, skupina modulu (umožňuje seskupovat moduly dohromady, což má za efekt, že moduly se budou v seznamech řadit za sebe), jméno a email autora modulu a „Action class“. Poslední parametr určuje Java třídu, která je volána při inicializaci, nebo modifikaci modulu. Tato třída umožňuje vykonávat

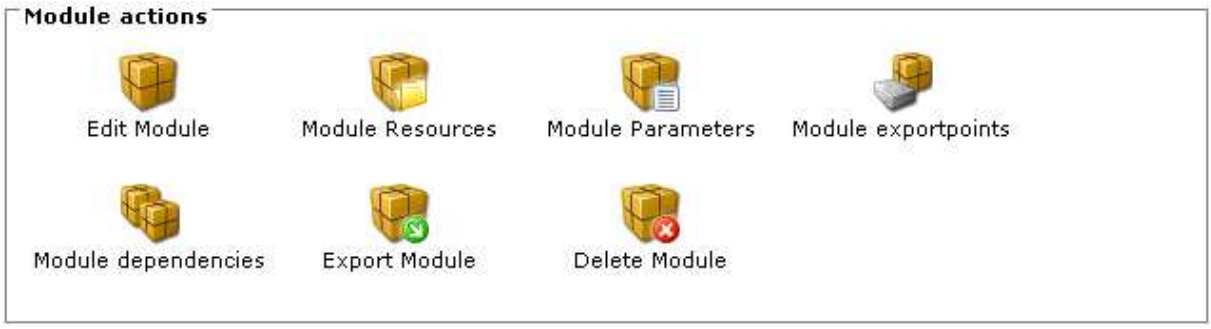
speciální funkce na základě podnětů událostí životního cyklu OpenCms (např. publikování projektu, ve kterém se nachází modul). „Action class“ musí implementovat rozhraní *org. opencms.module.I_CmsModuleAction*.

Všechny adresáře, které se nachází v podstromu hlavní složky modulu ve VFS jsou součástí modulu a budou se s modulem exportovat do RFS. Tyto modulové složky lze rozdělit na tři typy:

- **Systémové adresáře –** Vyžadují pevně určené jméno a nejdůležitějšími z nich jsou adresáře:
  - **classes -** systémová složka obsahující Java properties bundle soubory s nastavením a lokalizačními řetězci pro modul. V této složce je vytvořena úplná podadresářová struktura na základě názvu modulu.
  - **lib -** systémová složka obsahující JAR knihovnu určenou pro modul.
- **Konvenční adresáře –** Můžeme je pojmenovat libovolně, ale je vhodné je pojmenovat dle zavedených konvencí. Často budeme využívat adresáře:
  - **resources -** Složka, ve které uchovávám obrázky,CSS kaskádové styly a JavaScripty.
  - **elements -** Složka zahrnuje JSP soubory s XML ‘template‘ elementy.
  - **schemas -** Do této složky ukládám XSD schémata pro *XmlContent* typy resource objektu, které jsou součástí modulu.
- **Uživatelské adresáře –** Zbytek adresářů, které neodpovídá předešlým dvou lze pojmenovat libovolně.

Kompletní seznam systémových a konvenčních adresářů lze nalézt na [12].

K modulu se mohou navíc připojit libovolné resource objekty z celého VFS, jako „Module resources“ (viz Obr-3.4), které tak doplní resource objekty z podstromu složky modulu. To se může hodit, pokud potřebujeme do modulu zahrnout nově vytvořenou položku v administrační části Workplace, nebo pokud změníme nějakou funkčnost ve Workplace exploreru a rádi bychom jí distribuovali do jiného systému.



Obr-3.4 – Administrace modulu v Administration view

3.5.2 Modul v RFS

Všechny moduly v RFS jsou umístěny v adresáři *OPENCMS_RFS_FOLDER/WEB-INF/packages/modules* ve formě ZIP archivu. Pokud je modul exportován z VFS do RFS, pak právě do předešlé složky. Naopak pokud chceme modul importovat, nahrajeme jej do této složky, kde jej OpenCms dokáže lokalizovat.

ZIP archiv obsahuje všechny resource objekty, které patří k modulu a dále soubor *manifest.xml* s nezbytným systémovým nastavením modulu, zajišťujícím převod modulu z RFS do VFS při instalaci modulu do OpenCms.

V RFS lze moduly spravovat v konfiguračním souboru *opencms-modules.xml*. V něm se nachází informace o všech modulech nainstalovaných v systému.

3.6 Podpora procesu vývoje ze strany OpenCms

OpenCms nabízí poměrně rozsáhlé API pro přístup k VFS a jeho funkcionalitě na JSP stránkách a zároveň v Java třídách. Možnosti jak tento přístup získat jsou dvě. Použitím OpenCms taglib knihovny, nebo přes objekt třídy *org.OpenCms.jsp.CmsJspActionElement.*

3.6.1 OpenCms JSP taglib knihovna

OpenCms poskytuje pro přístup k některým běžně využívaným funkcím systému vlastní knihovnu značek, kterou je vhodné používat na JSP stránkách v systému. Tuto knihovnu lze také libovolně kombinovat s jinými JSP taglib knihovnami. Dobrou volbou pro kombinaci je zejména standardní knihovna značek JSTL v kombinaci s EL, které tvoří výrazné rysy technologie JSP 2.0.

Před použitím knihovny značek na JSP stránce je nutné ji nejprve deklarovat pomocí direktivy *taglib* (viz Src-3.1). Tím je knihovna jednoznačně identifikovaná a k jejím značkám lze přistupovat přes příponu *cms*.

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms" %>
```

Src-3.1 – deklarace knihovny značek

Dostupné jsou značky např. pro přístup k objektu aktuálně přihlášeného uživatele, k property položkám aktuálního resource objektu, k různým provozním informacím systému, značky pro procházení a prezentování strukturovaného obsahu typu *XmlContent* a zejména značky pro vytváření šablon v OpenCms.

3.6.1.1 Systém šablon v OpenCms

Šablony představují pro OpenCms další významný prvek pro usnadnění vytváření webových stránek. Šablona v OpenCms nese označení „template“ a nejedná se vlastně o nic jiného, než klasický resource objekt typu JSP. Šablonu z tohoto resource objektu činí jeho umístění ve VFS stromu. Za šablonu se resource objekt považuje, pokud se nachází v přímém podadresáři *templates/* uvnitř adresáře některého z modulů pod */system/modules/*. Pro takovou šablonu pak poskytuje funkčnost OpenCms knihovna značek a samozřejmě i samotné OpenCms API.

Šablona se může skládat z různých elementů, které dohromady mohou tvořit kompletní webovou stránku. Např. se může jednat o elementy s navigací, hlavičkou, patičkou, aktualitami apod. Elementy může šablona získávat ze stránek typu *xmlpage*, *xmlcontent*, ale i JSP stránka může být rozdělena na elementy pomocí značky `<cms:template>`. Ukázka jednoduché šablony je na Src-3.2. V této šabloně je vyznačené použití OpenCms knihovny značek. První vyznačení ukazuje již známou deklaraci knihovny značek, druhé vyznačení ukazuje zobrazení property položky *Title* aktuálního resource objektu a třetí vyznačení ukazuje vložení elementu *body* do této šablony. Element se v tomto případě bude brát z resource objektu, který šablonu využívá.

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms"%>
<html>
  <head>
    <title>
      OpenCms - <cms:property name="Title" escapeHtml="true" />
    </title>
  </head>
  <body>
    <cms:include element="body" />
  </body>
</html>
```

Src-3.2 – Ukázka jednoduché šablony v OpenCms

3.6.2 Přístup k OpenCms API přes CmsJspActionElement

Třída *CmsJspActionElement* představuje ekvivalent k OpenCms knihovně značek a ve skutečnosti dokonce oba způsoby využívají stejné interní metody. S touto třídou je však možné se dostat mnohem dál. Vytvořením její instance se totiž mohu dostat k další funkčnosti a správě, kterou nabízí OpenCms API.

OpenCms doporučuje tuto třídu používat ve formě JSP skriptletů na JSP stránkách a k tomu je třída také přizpůsobena. Její instanci lze vytvořit s pomocí objektů dostupných na každé JSP stránce:

- javax.servlet.jsp.PageContext,
- javax.servlet.http.HttpServletRequest,
- javax.servlet.http.HttpServletResponse,

jako parametrů konstruktoru. Skriptlety však mohou stránku dělat poměrně nepřehlednou a proto se jako lepší způsob jeví na JSP stránce pouze vytvořit instanci této třídy a další logiku spojenou s OpenCms API přesunout do Java tříd představující aplikační vrstvu. Nyní ale zpět k tomu, co nabízí *CmsJspActionElement.* Mimo funkcí totožných s OpenCms knihovnou značek totiž poskytuje přístup ke třídě *org.opencms.file.CmsObject.*

3.6.2.1 Třída CmsObject

CmsObject je ústřední třídou celého OpenCms API, poskytující plně autorizovaný přístup ke všem resource objektům ve VFS a všem operacím, které VFS nabízí pro práci s nimi. Navíc také zapouzdřuje objekt uživatele a jeho přístupová práva vzhledem k aktuálnímu resource objektu. K dispozici je i přístup ke třídě *org.opencms.file.CmsRequestContext,*

skladující informace o OpenCms kontextu aktuálně přihlášeného uživatele (např. lze zjistit URI HTTP požadavku, aktuálně zvolený projekt, *Site* položku a další).

3.6.2.2 Třída OpenCms

Třída *org.opencms.main.OpenCms* je singleton[2] a nabízí alternativu přístupu k funkcím OpenCms VFS pro třídu *CmsObject*. Rozdíl tvoří fakt, že tato třída poskytuje **statické** metody, které mohou být použity jinými třídami k přístupu do OpenCms API.

Třída *OpenCms* se také nazývá „operačním systémem" celého OpenCms, protože se přes ni lze dostat k ještě pokročilejším funkcím systému, než ze třídy *CmsObject*. Umožňuje měnit dokonce základní konfiguraci systému. Tento přístup poskytuje přes třídy zvané „Manager". Ty zaobalují funkce z určité související části OpenCms. Jako příklad manager třídy uvedu následující:

- *org.opencms.site.CmsSiteManagerImpl* – Spravuje všechny *Site* položky dostupné v OpenCms VFS.
- *org.opencms.security.CmsRoleManager* – Poskytuje přístup k operacím, které souvisí s rolemi v OpenCms.
- *org.opencms.module.CmsModuleManager* – Spravuje všechny moduly instalované v OpenCms.

[2] Singleton třída může mít jednu a právě jednu instanci.

4 Návrhy na vylepšení OpenCms

OpenCms je kvalitní systém pro správu obsahu, který se neustále vyvíjí díky aktivní komunitě lidí, kteří se tímto systémem zabývají ať už z obchodního, či ze studijního hlediska zájmu. Za dobu, kdy je OpenCms na „open-source trhu“ (od roku 2000) si vytvořil poměrně výrazné jméno, hlavně mezi Java open-source systémy pro správu obsahu na webu. Díky tomu počet jeho uživatelů a vývojářů rychle roste[1]. Samozřejmě však platí, že čím více je uživatelů systému, tím rozmanitější požadavky jsou na něj kladeny. Důležité pak je, že hlavní vývojáři ze společnosti *Alkacon Software GmbH* se snaží být k těmto požadavkům otevření a pracují s nimi. V tom také spočívá přínos open-source projektů.

Cílem této kapitoly je vytvořit souhrn aktuálních požadavků a návrhů na vylepšení OpenCms sebraných jak ze strany komunitou řízených zdrojů, tak ze strany katedry KIV. Výstup kapitoly pak bude tvořit konečný výběr některých z těchto požadavků, které budou v rámci této práce implementovány.

4.1 Komunita OpenCms

OpenCms je systém, za kterým stojí již poměrně rozsáhlá a aktivní komunita open-source vývojářů, v jejímž čele stojí společnost Alkacon Software GmbH, která se stará o udržování plánu práce na tomto systému, a zejména o poskytování podpory pro open-source vývojáře.

K dispozici nabízí podporu ve formě následujících zdrojů:

- **opencms-dev mailing list [9]** – Místo pro otázky a komentáře týkající se nejen vývoje v OpenCms. V roce 2008 zde bylo registrováno kolem 1500 odběratelů a přispěvatelů. K odběru se lze přihlásit na stránce mailing listu, kde je dostupný i rozsáhlý archiv příspěvků sahající až do roku 2000.

[1] Oficiální prameny prezentované na konferenci *OpenCmsDays2008,* dostupné na [11] uvádí, že na oficiální stránky OpenCms denně zavítá více než 2000 návštěvníků a počet stažení aktuální verze OpenCms7 se pohybuje okolo čísla 5000 stáhnutí za měsíc (v průměru 150-200 denně).

- **Alkacon documentation [9] + demo aplikace** – Dokumentace zahrnuje přehledně strukturovaný soubor návodů pro vývojáře začátečníky i pokročilé. Její součástí je také Javadoc dokumentace k OpenCms API jádra systému. Dokumentace je dostupná online na oficiálních stránkách pro vývoj OpenCms, nebo je také volitelně součástí standardní instalace OpenCms. Se standardní instalací navíc můžete získat i ukázkové demo aplikace, ze kterých se lze naučit princip vývoje webových stránek v OpenCms.
- **OpenCms Wiki** – Wiki stránky [10] řízené přímo ze společnosti Alkacon software GmbH. V současné době je zde k dispozici okolo 180 článků týkajících se OpenCms.

Dále existuje řada diskusních fór zaměřených na OpenCms, z nichž nejrozsáhlejší a nejaktivnější je fórum na [18].

Alkacon Software GmbH se také podílí z 97% na vývoji jádra systému. U vývoje nových modulů však poskytuje větší prostor vývojářům z organizací, které používají OpenCms a vývojářům od oficiálních poskytovatelů OpenCms řešení, mezi něž patří i česká společnost *Qbizm Technologies, Inc. (www.qbizm.com).* Pokud někdo z těchto vývojářů udržuje sadu modulů, řádně otestovaných v provozu a splňujících určitá kritéria, pak může být tato sada zařazena do nadstavbové sady modulů Alkacon OAMP, které sice nejsou standardní součástí OpenCms, ale jsou volně dostupné na oficiálních stránkách firmy Alkacon [14].

Alkacon také průběžně aktualizuje plán budoucího vývoje OpenCms na wiki stránkách. V něm zveřejňuje „Release plan“, zahrnující plán vývoje funkčností vzhledem k budoucím verzím OpenCms a dále „Wishlist“, neboli seznam nejčastějších požadavků na rozšíření OpenCms. Od tohoto seznamu pak očekává, že by se v implementaci jeho položek mohli angažovat právě poskytovatelé řešení OpenCms a organizace užívající OpenCms.

4.1.1 OpenCms „Roadmap“

Na konferenci *OpenCms Days 2008* (prezentace dostupná na oficiálních stránkách OpenCms [11]) bylo přislíbeno vydání následujících verzí OpenCms v těchto termínech:

- Verze 7.5 – červen 2009.
- Verze 8.0 – červenec 2010.

4.1.1.1 Předběžný plán Verze 7.5

Verze by měla již využívat Java genericitu. Dále bude zahrnovat následující funkčnosti:

- **Vylepšení manipulace s obrázky v editorech**

  - Podpora pro oříznutí obrázku z galerie.
- **Vylepšení editorů pro *XmlContent.***
  - Přidání podpory pro `xsd:choice`.
  - Více využívání technologie Ajax.
- **Vylepšení použitelnosti prvku *Direct Edit.***
  - Uspořádání položek na stránce pomocí Drag & Drop.
  - Podpora přímého vkládání a mazání resource objektů.
  - Manipulace s navigací.
- **Rozšíření funkcionality EL.**

4.1.1.2 Předběžný plán verze 8.0

Předpokládá se využití Java JDK 6 a vyšší, rozšíření API o podporu webových služeb (JAX-WS) a integrace skriptovacího jazyka. Dále by měla verze zahrnovat následující funkčnosti:

- **Podpora JSR 170 –** JSR 170 je standardizované API pro přístup k úložištím.
- **Podpora přístupu do VFS přes webové služby**
- **Integrace externího workflow enginu –** Nejspíše enginu JBoss jBpm.
- **Využití JPA (Java persistence API) pro persistenci dat do DB.**
- **Přístup do OpenCms API pomocí JavaScript.**

4.1.2 OpenCms „Wishlist“

Udržuje další požadavky na vylepšení OpenCms, které se nevešly do plánů dosud avizovaných verzí. Mezi požadavky patří:

- **Oddělení produkčních dat od nasazených souborů v RFS –** Neoddělení neumožňuje využití standardních J2EE procedur (např. redeploy WAR balíků při změně), a to činí rozšíření OpenCms o nové funkčnosti velice nepohodlným.
- **Možnost definovat pro každý modul vlastní uživatelský zavaděč –** Díky tomu by nebylo potřeba exportovat resource objekty jako Java třídy a JAR balíky z VFS do RFS a tím pádem by změna v těchto souborech nevyžadovala restartování OpenCms.
- **Podpora pro vytvoření *manifest.xml* vně systému OpenCms –** Tato položka by umožnila vytvářet moduly v RFS.
- **Zlepšit zvýraznění syntaxe JSP souborů ve VFS Workplace editorech.**

- **Zlepšit podporu pro JSF –** JSF 1.1 je již v OpenCms podporováno. Verze JSF 1.2 však nefunguje podle očekávání.

4.2 OpenCms na katedře KIV

Na katedře KIV se využívá OpenCms od roku 2006, kdy jej v rámci své DP (viz [4]) vybral Ing. Josef Krupička jako CMS, na kterém bude postaven web katedry. Zprovoznění testovací verze webu a transformace ostré verze na OpenCms pak proběhla v roce 2008.

4.2.1 Náměty na vylepšení OpenCms

Za dobu používání OpenCms se již na KIV a CIV stačilo posbírat několik nápadů na rozšíření funkčnosti systému a jejich seznam je zveřejněn a aktualizován na wiki stránkách KIV, věnovaných OpenCms [19].

4.2.1.1 Náměty s vysokou prioritou

- **Záložky ve workplace exploreru –** Možnost přidávat do záložek resource objekty, tak aby k nim byl rychlejší přístup ve Workplace. Zároveň by toto vylepšení mohlo zahrnovat uchování stavu aktuální session při odhlášení uživatele a její opětovné nahrání při opětovném přihlášení stejného uživatele. Záložky by mohli být generovány i automaticky (např. seznam záložek naposledy modifikovaných resource objektů).
- **Report s přehledem přístupových práv ve VFS podstromu –** Možnost zobrazit souhrn přístupových práv vybraného uživatele/skupiny v celém stromu VFS od kořenového adresáře, který bude volitelný.
- **Úprava základního property dialogu –** Účelem je zpřehlednění dialogu a editace property položek na resource objektu. To docílíme parametrizací základního property dialogu na základě použité šablony, nebo typu resource objektu. Zároveň bude třeba zjednodušit zadávání jednotlivých property položek.
- **Možnost vyřadit modul z provozu ve VFS –** Modul zůstane nainstalovaný ve VFS, avšak všechny resource objekty, které zaobaluje, již nebudou k mání.
- **Vylepšit záznam o změnách resource objektů –** Vylepšit záznam udávající kdo, kdy, co a jak změnil/udělal ve VFS, který částečně implementuje položka administrační části Workplace *Project Management -> Project history.*
- **Vylepšit proces publikace –** Umožnit přidat komentář k publikování, tak aby byl viditelný i v historii. Uživatelům, kteří nemají právo přímo publikovat resource

objekty, by mohlo být umožněno alespoň navrhnout resource objekt na publikaci. S tím by pak byl využitelný dialog pro volbu resource objektů, které mají být publikovány.

4.2.1.2 Další náměty na vylepšení

Součástí jsou náměty s normální prioritou z [19], dále pak různá vylepšení, která vyplynula z hledání inspirace u jiných systémů pro správu obsahu v rámci této diplomové práce. Mezi systémy, které jsem procházel, patří systémy Xaraya, Typo3, Alfresco a Magnolia. Zahrnuty jsou též návrhy z [17], které jsou založené na systémech CoreMedia, RedDot a Day.

- **Česká lokalizace Workplace**
- **Vylepšení týkající se Workplace exploreru**
  - **Branding workplace –** Do workplace přibude banner s výraznou identifikací webové stránky.
  - **Možnost zobrazit náhled obsahu ve workplace –** Po najetí na resource objekt ve Workplace exploreru by se zobrazilo vyskakovací okno s náhledem stránky i vybranými meta informacemi.
  - **Možnost filtrovat a řadit resource objekty ve Workplace exploreru –** Bude možné zobrazit pouze určitý typ resource objektů, popř. volba řadit resource objekty podle jména, typu resource objektu, velikosti, času poslední modifikace apod.
  - **U resource objektů ve Workplace exploreru zobrazovat více informací –** Nabízí se např. jméno autora resource objektu. Větší efektivitu by mohlo přinést i přesunutí nejběžnějších akcí z kontextového menu resource objektu přímo do seznamu resource objektů. (např. publish directly, touch).
- **Vylepšení týkající se WYSIWYG editoru**
  - Zobrazit aktuální stav publikace editovaného resource objektu (žlutá vlajka) a ve spojitosti s tímto nabídnout přímo v editoru funkci „publish directly“.
  - Přidat do editoru tlačítkovou lištu s pokročilými funkcemi nad resource objektem (zobrazení property dialogu, historie, nastavení přístupových práv apod.)
- **Vylepšit logování OpenCms –** Zachytit, co se děje uvnitř jádra OpenCms.
- **Zjednodušit nastavení přístupových práv na resource objektech –** Standardně lze přístupová práva pro resource objekty nastavovat pouze v konfiguračních souborech v

RFS. Bylo by dobré poskytnout také možnost nastavení v administrační části Workplace.

- **Vytvořit archiv pro smazané resource objekty –** Udržovat v seznamu volitelný počet naposledy smazaných resource objektů a umožnit nad nimi provést akci „Obnovit“.

4.3 Závěr

Možností k rozšíření by se naskytlo opravdu hodně, proto nebylo jednoduché určit pouze malou množinu z nich, která bude implementována. Již ze začátku tedy padlo rozhodnutí, že vybraná rozšíření se budou vybírat pouze z námětů s vysokou prioritou sesbíraných na KIV, protože ty vzešly přímo z určitých případů užití na webu katedry.

Na základě diskuze s Ing. Přemyslem Bradou, MSc., Ph.D a Ing. Josefem Krupičkou nad jednotlivými návrhy a s přihlédnutím k rozsahu této práce, byly k implementaci zvoleny dva následující:

- **Úprava základního property dialogu**
- **Report s přehledem přístupových práv ve VFS podstromu**

Jejich implementace bude popsána v následující kapitole.

5 Implementace rozšíření OpenCms

Tato kapitola popisuje implementaci dvou rozšíření OpenCms, vylepšující funkčnosti jádra systému, vybraných v kapitole 4. Jedná se o rozšíření „Parametrizovaný property dialog“ a rozšíření „Report s přístupovými právy“. Nejprve se zde zaměřím na způsob distribuce nových rozšíření a popis jejich architektury, která bude pro obě společná. Následně postupně pro obě rozšíření zmíním původní stav řešeného problému v OpenCms, popis analýzy a detaily implementace rozšíření. Nakonec uvedu informace o nasazení modulů do provozu.

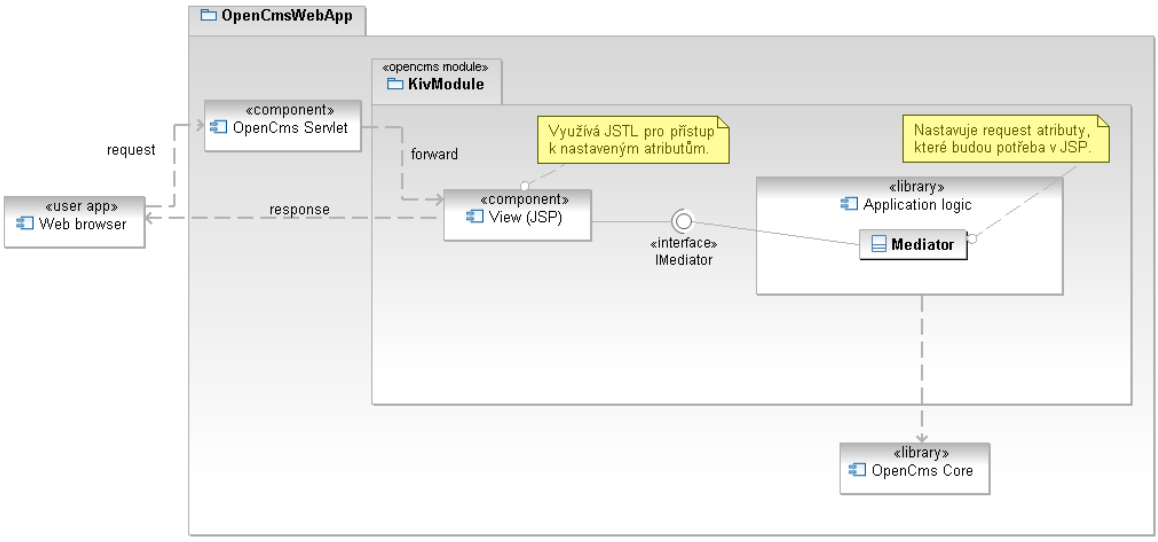
5.1 Architektura implementace

Aby byla implementovaná rozšíření snadno aplikovatelná na různé instalace OpenCms, rozhodl jsem se pro každé z nich vytvořit samostatný modul, který bude zaobalovat veškerou jejich funkčnost. Tyto moduly budou dodržovat doporučenou strukturu modulů v OpenCms, a všechny tedy budou zahrnovat minimálně „systémové“ a „konvenční“ adresáře zmíněné v kapitole 3.5.1.

5.1.1 Mediator pattern architektura

Mediator pattern je součástí „The Gang of Four“ (GOF) návrhových vzorů, které rozdělují tyto vzory do skupin. Mediator patří do skupiny „Vzory chování“.

Jedná se o architekturu aplikovanou na nově vznikající moduly na KIV kvůli potřebě oddělit v nich view a logiku. Dále tato architektura řeší i nesnadnou integrace Java Servletů do OpenCms modulů. Hlavní prvky architektury – *Mediatory* částečně tyto Servlety nahrazují.



Obr-5.1 – Diagram komponent architektury „Mediator pattern“.

Architektura si nijak neodporuje s doporučenou strukturou modulu z kapitoly 3.5.1, spíše na ni navazuje. Komponent diagram Mediator architektury modulu je znázorněn na Obr-5.1.

5.1.1.1 Mediator

Původně byly Mediatory navrženy jako prostředníky mezi JSP stránkami z OpenCms modulů a Spring DAO objekty z datové vrstvy pro moduly. Je vhodné je využít ale i v modulech, které Spring datovou vrstvu neobsahují. *Mediatory* totiž samy o sobě tvoří *Controller* vrstvu pro *view* reprezentované JSP stránkami modulů.

Mediator je definován rozhraním *IMediator*, které zahrnuje *handle(...)* metody, které jsou analogií *doGet()* metody u Java Servletu. Příkladem *handle(...)* metod mohou být následující:

- `handle(PageContext, HttpServletRequest, HttpServletResponse)`
- `handle(HttpServletRequest, HttpServletResponse)`
- `handle(CmsJspActionElement)`

V konkrétní třídě Mediatoru pak stačí předefinovat alespoň jednu z nabízených metod, tak, že ji oddědím od abstraktní třídy *AbstractMediator*, implementující toto rozhraní. Mediator v *handle* metodě zpracovává požadavky, jdoucí z JSP stránek. Dále může poskytovat jako Controller další logiku a funkčnost pro moduly. Výsledky zpracování předává na „view“ vrstvu ve formě nastavených atributů implicitních objektů.

Mediator poskytuje i přístup k funkcionalitě, kterou nabízí OpenCms. To je zařízené tím, že *AbstractMediator* je oděděn od OpenCms bean třídy *org.opencms.jsp. CmsJspActionElement*, která umožňuje pohodlný přístup k funkcionalitě spojené s jádrem OpenCms a s VFS. *CmsJspActionElement* dává také k dispozici metodu `getCmsObject()`, díky které se dostaneme ke stěžejní třídě Opencms *org.opencms.file.CmsObject*, poskytující plně autorizovaný přístup k VFS resource objektům.

5.1.1.2 View vrstva

Je reprezentována JSP stránkami v modulech. Stránky na svém začátku vytvoří objekt asociovaného Mediatoru a zavolají vhodnou metodu *handle*, viz Src-5.1.

```
<%
  IMediator mediator = new CustomMediator();
  mediator.handle(pageContext, request, response);
%>
```

Src-5.1 – Skriptlet na view vrstvě volající mediator

Díky tomu, že Mediator obstarává veškerou logiku, zůstane skriptlet ze Src-5.1 jediným na JSP stránce. Ve zbytku JSP stránky se bude využívat JSTL pro práci s nastavenými request atributy.

5.1.2 Lokalizace modulů

Lokalizované L10N řetězce se ukládají do adresáře modulu *classes* do Java properties bundle. K těmto řetězcům pak přistupuji dvěma různými způsoby.

1. Na JSP stránkách používám JSTL sadu tagů z *fmt* knihovny.
2. V Java třídách využívám rozhraní *org.opencms.i18n. I_CmsMessageBundle*. Pro podporu lokalizovaných řetězců ve vlastních Java třídách stačí v rámci modulu poskytnout podtřídu třídy *org.opencms.i18n.A_CmsMessageBundle*[1]. Tato podtřída by se podle konvence měla jmenovat vždy *Messages.java* a má za úkol číst lokalizované řetězce z odpovídajícího message bundle[2], jehož název je deklarován staticky uvnitř třídy.

[1] Abstraktní třída *A_CmsMessageBundle* implementuje rozhraní *I_CmsMessageBundle.*

[2] Podle konvencí se message bundle pojmenovává jménem *message.properties.*

5.2 Modul „Parametrizovaný property dialog“

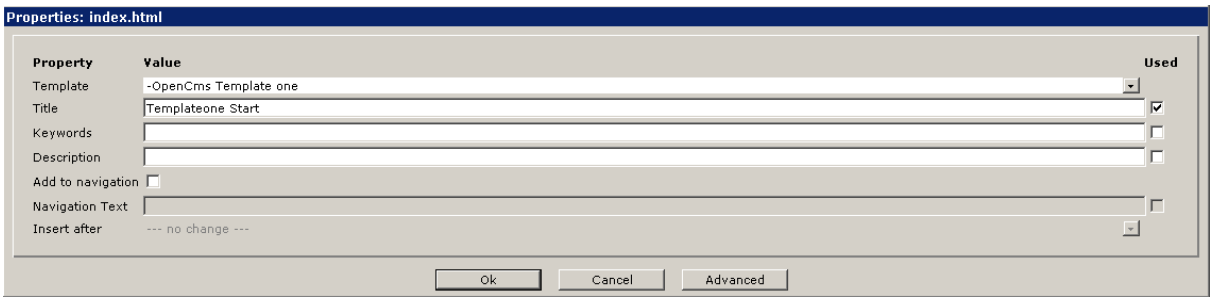
Pro každý resource objekt v OpenCms je možné konfigurovat speciální sadu metadat. Tato sada se nazývá ‘*properties*‘ a je konfigurovatelná přes property dialog vyvolaný z kontextového menu resource objektu. Pro všechny resource objekty ve VFS je tato sada stejná, je totiž definována pro celý systém s tím, že lze definovat novou property položku a samozřejmě odstranit již definovanou. Jedna property položka představuje dvojici *název/hodnota*.

5.2.1 Původní stav property dialogu v OpenCms

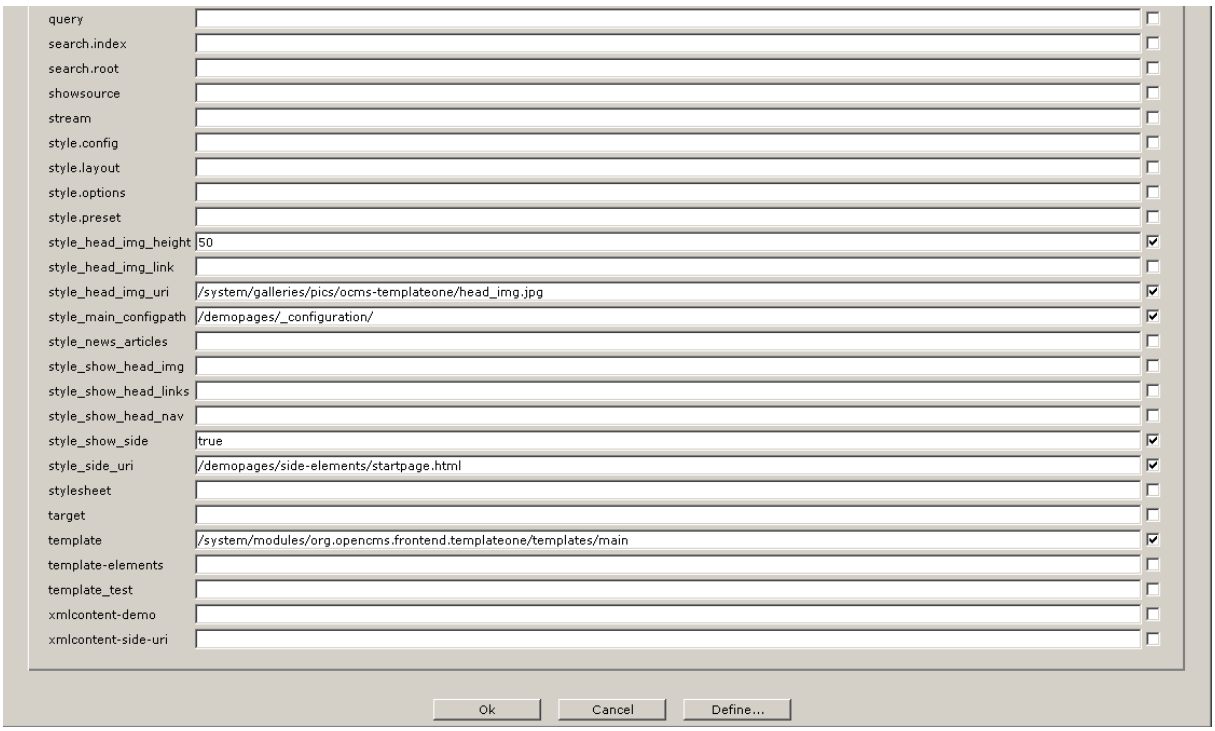
Nejprve se pozastavím nad standardním property dialogem, který OpenCms nabízí pro editaci property položek. Property dialog u každého má k dispozici dva základní pohledy:

- **Základní (basic) pohled** – Na tento pohled se dostanu z kontextového menu resource objektu volbou *Properties* a jeho prostřednictvím je možné editovat pouze vybrané property položky. Standardně jsou v OpenCms dva základní property dialogy – jeden pro typ resource objektu *xmlPage* a druhý pro ostatní typy resource objektů. Liší se pouze tak, že dialog pro *xmlPage* obsahuje vysunovací nabídku (*SelectWidget)* pro volbu property položky *template*. Ostatní hodnoty se editují prostřednictvím textových polí. Dále se může ještě výběr položek pro základní dialog lišit v závislosti na *explorer type settings* aktuálního typu resource objektu.[3] Základní pohled property dialogu ilustruje Obr-5.2.
- **Pokročilý (advanced) pohled** – Na tento pohled se dostanu pomocí tlačítka *Advanced* ze základního pohledu a jeho prostřednictvím je možné editovat všechny property položky definované v OpenCms. Hodnoty k položkám lze zadávat jen jako text do textového pole (*StringWidget*). Na konci dialogu se nachází tlačítko *Define,* přes které je možné otevřít dialog pro definování nové property položky. Část pokročilého pohledu property dialogu ilustruje Obr-5.3.

[3] *Explorer type settings* jsou definovány v konfiguračních souborech *opencms-modules.properties* a *opencms-workplace.properties* v části <*explorertypes*> poskytující nastavení pro typy resource objektu v OpenCms VFS. V *opencms-modules.properties* se vyskytují nastavení pro *XmlContent* typy definované v rámci modulů a v *opencms-workplace.properties* se vyskytují nastavení pro základní typy v OpenCms. Základní property položky lze resource typu přiřadit v elementu <*explorertype*> -> <*defaultproperties*>.



Obr-5.2 – Základní pohled původního property dialogu pro *XmlPage* typ resource objektu.



Obr-5.3 – Část pokročilého pohledu původního property dialogu pro *XmlPage* typ resource objektu.

5.2.2 Analýza problému

Pro běžného uživatele editora nabízí původní property dialog dva různé pohledy. Jak se ukázalo, ani jeden není nejlepším efektivním řešením. Editace přes pokročilý pohled je velice nepřehledná, zvláště když počet property položek v OpenCms má tendenci stále narůstat s novými stránkami, šablonami i typy resource objektů. I s novými moduly přichází do systému nové property položky. Ty jsou pak použitelné prakticky jen pro modul, typ resource objektu, nebo šablonu, ke kterým byly původně definovány a u ostatních typů resource objektu působí pouze jako rušivý element. Základní dialog pro změnu nabízí pouze nejzákladnější sadu property položek nutnou pro každý resource objekt, přitom by se nabízela možnost editovat v základním dialogu property položky specifické pro typ resource objektu,

nebo šablonu, jíž aktuální resource objekt využívá. Základní dialog by také měl být co nejvíce přístupný všem uživatelům a nabízet k editaci property položek adekvátní widget prvek (ne jenom textové pole).

Idea pro tento modul je taková, že většina uživatelů by si měla vystačit při editaci property položek se základním dialogem, nabízejícím pestrou paletu widget prvků a omezujícím uživatele v tom dobrém slova smyslu. Pokročilý dialog by zůstal dále stejný a měl by vlastně působit jen jako nástroj zachovávající pocit svobody editace property položek, přes který bude stále možné zadat k property položce jakoukoli hodnotu, která v případě nouze bude upřednostněna před hodnotou ze základního dialogu.

5.2.2.1 Požadavky na rozšíření

Hlavním úkolem tohoto modulu je poskytnout uživatelům OpenCms parametrizovaný property dialog na resource objekt s následujícími možnostmi:

- Definovat editovatelné property položky z množiny všech položek v systému přímo v základním property dialogu.
  - Základní property dialog bude konfigurovatelný.
- Navrhnout konfiguraci základního property dialogu u resource objektu, tak aby byla závislá na šabloně, kterou resource objekt používá, popř. na jeho typu.
- Pro jednotlivé property položky mít možnost nadefinovat (v konfiguračním souboru) widget prvky použité k jejich editaci.
  - Widget prvky bude možné vybrat z předdefinované sady.

Další úpravy kosmetického rázu jsou následující:

- Auto focus na první položku ve formuláři property dialogu.
- Automaticky kopírovat hodnotu property položky *Title* do property položky *NavText (navigační text)*, pokud ta nemá vyplněnou hodnotu.

5.2.2.2 Architektura property dialogu v OpenCms

K tomu, abychom mohli přejít k řešení problému, je nejprve nutné popsat, jakým způsobem funguje v OpenCms standardní property dialog.

OpenCms dokáže rozpoznat standardně pouze tři různé typy dialogů ve workplace:

- Delete dialog,
- Lock/Unlock dialog,

- File properties dialog.

V kontextu tohoto modulu nás bude zajímat třetí typ dialogu. Nejprve je třeba ale uvést, kde se nachází logické jednotky (Java třídy, JSP), které tyto dialogy obsluhují. Handlery pro každý z těchto dialogů jsou konfigurovatelné a tato konfigurace se nachází v souboru *OPENCMS/WEB-INF/config/opencms-workplace.xml*. Tento soubor obsahuje mimo jiné i element `<dialoghandlers>` se třemi potomky. Každý z těchto potomků je dialog handler právě pro jeden ze tří typů dialogů a důležité je, že na tomto místě mohou být vždy pouze tři potomci – pro každý typ dialogu pouze jeden handler. Pokud tedy bude potřeba vytvořit uživatelský handler pro jeden z těchto dialogů, musíte jím nahradit ten stávající. Ukázka konfigurace je naznačena na Src-5.2.

```
<dialoghandlers>
  <dialoghandler class="org.opencms.workplace.commons.CmsDelete"/>
  <dialoghandler class="org.opencms.workplace.commons.CmsLock"/>
  <dialoghandler class="org.opencms.workplace.commons.CmsPropertyAdvanced"/>
</dialoghandlers>
```

Src-5.2 – Konfigurace property dialog handlerů.

Jak lze vyčíst ze Src-5.2, default property dialog handler v OpenCms je třída *org.opencms.workplace.commons.CmsPropertyAdvanced*. Tato třída implementuje rozhraní *org.opencms.workplace.I_CmsDialogHandler*, které donutí handler implementovat následující metody:

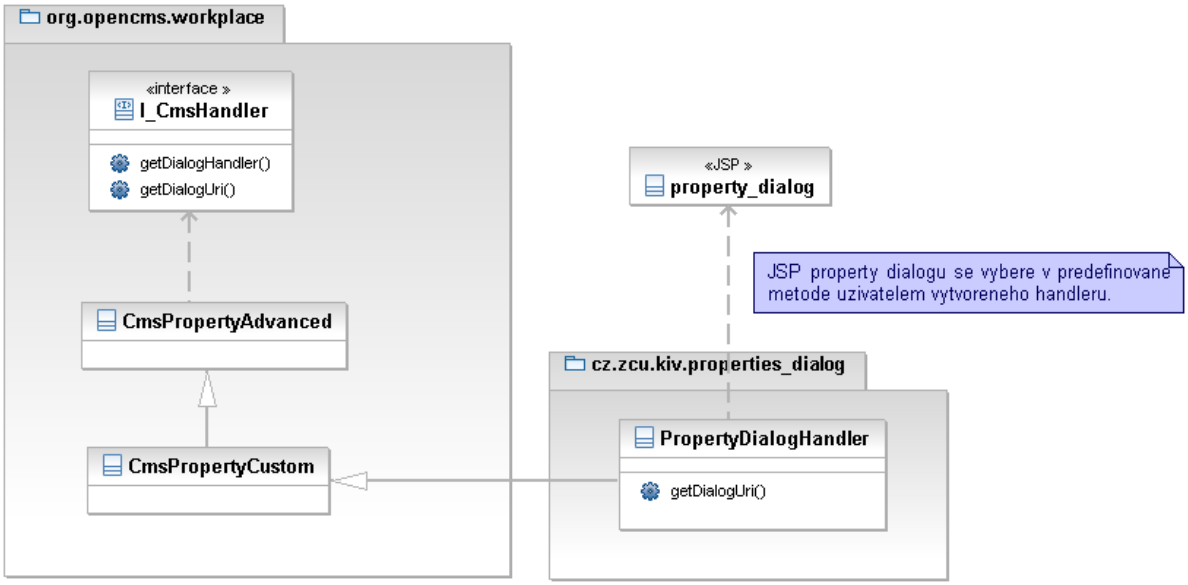
- `String getDialogHandler()` – vrací jméno dialog handleru.
- `String getDialogUri(String resource, CmsJspActionElement jsp)` – vrací absolutní cestu k property dialogu vybraného resource objektu, který je reprezentován JSP souborem, zobrazujícím HTML kód dialogu. Tato metoda bude využita třídou, která se nazývá *Selector* k přesměrování na URI vybraného property dialogu.

Nyní již tedy známe metodu, ve které se rozhoduje, jaký property dialog bude použit. Zbývá zjistit, kdy se tato metoda volá. Zvolíme-li ve workplace možnost zobrazit property dialog na vybraný resource objekt, vyvolá se JSP z VFS *system/workplace/commons/property.jsp*, které provede dvě akce:

1. Vytvoří instanci třídy *org.opencms.workplace.CmsDialogSelector*. *Selector* na základě konfiguračního souboru vybere dialog handler třídu, které pak předá zodpovědnost najít absolutní cestu k property dialogu[4].
2. Vloží na stránku property dialog (jehož URI byl vybrán *Selectorem)* pomocí akce `cms.include`.

Pokud budeme vytvářet vlastní handler pro property dialog, využijeme k tomu třídu *org.opencms.workplace.commons.CmsPropertyCustom*, kterou oddědíme ve vlastním handleru (viz Obr-5.4). OpenCms tuto třídu poskytuje pro vytváření uživatelsky přizpůsobitelných property dialogů. *CmsPropertyCustom* je odděděna od výše zmíněného standardního handleru *CmsPropertyAdvanced*, takže nám bude stačit ve svém handleru překrýt metodu `String getDialogUri(String resource, CmsJspActionElement jsp)`, ve které si již můžeme naprogramovat vlastní logiku, podle které se bude volit property dialog.

Nakonec nahradíme v konfiguračním souboru původní property dialog handler tím vlastním, kterému poté (a po restartu webového kontejneru) *Selector* přenechá výběr property dialogu.



Obr-5.4 – Diagram tříd property dialog handleru v OpenCms.

[4] Ve většině případů by měl být property dialog reprezentován JSP souborem ve VFS.

5.2.3 Řešení problému

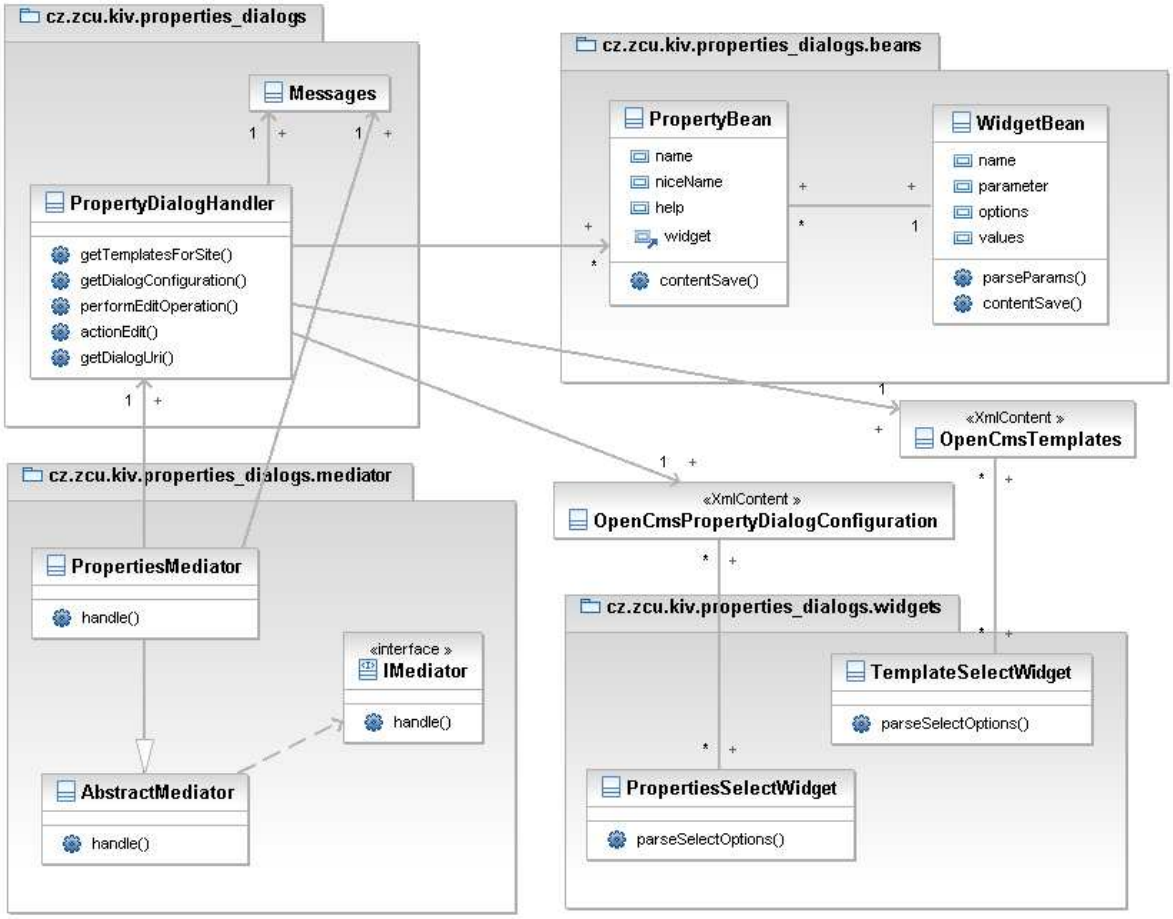
Nejprve popíši strukturu samotného modulu. Následně se budu věnovat vlastnímu property dialog handleru, který nahradí původní handler v OpenCms. Nakonec vás seznámím s možnostmi konfigurace property dialogu, včetně použití widget prvku pro property položky a zpracování konfiguračního souboru v property dialog handleru.

5.2.3.1 Struktura modulu

Název modulu je *cz.zcu.kiv.properties_dialogs*. Modul zahrnuje následující adresáře:

- Systémové adresáře **classes** a **lib** popsané v kapitole 3.5.1.
- Doporučené adresáře **elements, resources, schemas** popsané v kapitole 3.5.1. Složka **schemas** obsahuje XSD schémata pro konfigurační soubory property dialogu a pro konfigurační soubor přiřazující šablony k *Site*.
- **config** – Složka určená pro konfigurační soubory property dialogů spojené se systémovými typy resource objektu v OpenCms.
- **dialogs** – Zahrnuje JSP s vlastním property dialogem.

Modul obsahuje JAR knihovnu *cz.zcu.kiv.properties_dialogs.jar* v adresáři */lib*, jejíž součástí je sada Java balíků, které nyní ve stručnosti popíšu. Diagram tříd knihovny znázorňuje Obr-5.5.



Obr-5.5 – Diagram tříd z knihovny modulu *cz.zcu.kiv.properties_dialogs.jar*

Package cz.zcu.kiv.properties_dialogs

Základní balíček modulu zahrnuje třídu handleru property dialogu, jejíž název je *PropertyDialogHandler.java* a dále třídu *Messages.java* poskytující přístup k lokalizovaným řetězcům modulu viz kapitola 5.1.2.

Package cz.zcu.kiv.properties_dialogs.beans

Balík obsahuje Java Bean třídy ke *XmlContent* souborům s konfigurací property dialogu. Uvnitř se nachází dvě třídy:

- *PropertyBean.java*
- *WidgetBean.java*

Podrobnosti k těmto třídám budou popsány v části konfigurace property dialogu v kapitole 5.2.3.3.

Package cz.zcu.kiv.properties_dialogs.mediator

Balík obsahuje třídy, které jsou součástí tzv. “Mediator pattern“ architektury popsané v kapitole 5.1.1.

V tomto modulu používám jediný Mediator *PropertiesMediator.java*, který představuje prostředník mezi JSP property dialogem a property dialog handlerem. Jeho úkolem je předávat mezi dialogem a handlerem parametry ve formě atributů implicitních objektů. S pomocí těchto parametrů, přicházejících z JSP vytvoří handler požadované HTML pro dialog. JSP dialog pak využívá JSTL pro čtení parametrů a vyhne se tak nepřehledným skriptletům.

Package cz.zcu.kiv.properties_dialogs.widgets

Balík obsahuje dva uživatelské widget prvky, které jsou využité v rámci tohoto modulu. Jedná se o:

- *PropertiesSelectWidget.java* – Select widget prvek pro výběr jedné property položky ze všech definovaných v OpenCms. Widget prvek je použit v konfiguračním souboru property dialogu pro volbu položky.
- *TemplatesSelectWidget.java* – Select widget prvek pro výběr jedné šablony ze všech definovaných v OpenCms, s vyloučením *.cfg* šablon, představujících šablony pro property dialogy. Widget prvek je použit v konfiguračním souboru definujícím seznam šablon pro aktuální větev ve VFS podstromu viz kapitola 5.2.3.3.

5.2.3.2 Property dialog handler

Property dialog handler *PropertyDialogHandler.java* je stěžejní třídou celého modulu. Jak vyplývá z analýzy problému (viz kapitola 5.1.1), tento handler nahrazuje původní property dialog handler v OpenCms a přebírá tím zodpovědnost za výběr JSP, které představuje property dialog k aktuálnímu resource objektu. Ve stručnosti, tato třída obstarává mimo výběru property dialogu ještě veškeré akce nad property dialogem, jako uložení nové hodnoty property položky, uzavření dialogu bez uložení, další akce poskytnuté původním

property dialog handlerem[5] a dále pak vytvoření HTML pro samotný formulář property dialogu i s jednotlivými widget prvky a zpracování konfiguračního souboru. Ve skutečnosti tedy nevytváří přímo HTML pro formulář, ale pouze nastavuje přes *PropertiesMediator.java* potřebné atributy pro formulář, jehož HTML je vytvořeno ve view vrstvě v JSP souborech.

Výběr property dialogu

Jak bylo řečeno v kapitole 5.1.1.1, zobrazení vlastního property dialogu bude závislé na konfiguračním souboru, který bude definován na základě šablony, kterou aktuální resource objekt používá, nebo na základě jeho typu.

Volbu property dialogu obstarává metoda `String getDialogUri(String resource, CmsJspActionElement jsp)`, kde parametr `resource` představuje URL k resource objektu, na kterém je zobrazen property dialog.

Metoda vybírá ze tří property dialogů:

1. */system/modules/cz.zcu.kiv.properties_dialogs/dialogs/property_dialog.jsp* – dialog vytvořený v rámci tohoto modulu.
2. */system/workplace/editors/dialogs/property.jsp* – default property dialog pro *xmlPage* typ resource objektu.
3. */system/workplace/commons/property_custom.jsp* – default property dialog.

Property dialog se vybírá podle předchozího seznamu v pořadí, v jakém je v onom seznamu uveden. Až poté, když nejde použít v jednom případě, zkusí se použít dialog z následujícího bodu. V posledním případě bude vždy použit alespoň dialog z bodu 3, který se liší od dialogu č. 2 pouze v tom, že typ jeho resource objektu není *xmlPage*.

První property dialog bude použit, pokud k aktuálnímu resource objektu, na kterém je dialog zobrazen, existuje adekvátní konfigurační soubor[6], definující vzhled property dialogu. Ten se hledá podle následujících kritérií (v tomto pořadí):

[5] *PropertyDialogHandler.java* má přístup ke všem metodám jako původní property dialog handler *org.opencms.workplace.commons.CmsPropertyAdvanced*, protože je jeho nepřímým potomkem (přes třídu *org.opencms.workplace.commons.CmsPropertyCustom*).

[6] Konfigurační soubor bude popsán podrobněji v kapitole 5.2.3.3. Nyní pouze zmíním, že se jedná o *XmlContent* resource s příponou '*.cfg*'.

a) Podle OpenCms šablony, kterou má aktuální resource objekt přiřazenou pro zobrazení obsahu. Konfigurační soubor musí být ve VFS ve stejné složce jako použitá šablona, musí mít stejné jméno jako šablona (bez přípony souboru) a příponu *'.cfg'*.
b) Podle *XmlContent* typu aktuálního resource objektu. Konfigurační soubor musí být ve stejném adresáři jako XML schema soubor pro tento *XmlContent*, musí mít stejné jméno jako schema (bez přípony souboru) a příponu *'.cfg'*.
c) Podle jiného typu než *XmlContent*. Zde rozlišuji jiný typ a *XmlContent* typ z toho důvodu, že *XmlContent* je definován XML schématem umístěným uvnitř modulu, který jej zaobaluje a mohu tedy umístit konfigurační soubor pro property dialog k tomuto schématu. Jiný typ resource objektu toto schéma nemá ve VFS dostupné. Konfigurační soubor tedy musí být umístěn v adresáři */system/modules/cz.zcu.kiv.properties_dialogs/config/*, musí mít stejné jméno jako typ resource objektu[7] a příponu *'.cfg'*.

Pro lepší pochopení ještě uvedu příklad. Pokud má aktuální resource objekt přiřazenou šablonu, která nemá konfigurační soubor, pokusí se dialog handler hledat konfigurační soubor podle svého typu. Pokud ho nenajde ani v tomto případě (ať už se jedná o *XmlContent*, či ne), použije pro zobrazení property dialogu soubor z bodu 2 (když se jedná o *xmlPage*), nebo soubor z bodu 3 (když se jedná o jiný typ než *xmlPage*).

5.2.3.3 Konfigurační soubor pro property dialog

Pro vytvoření konfiguračního souboru property dialogu jsem využil možnosti strukturovaného obsahu v OpenCms a zvolil jsem tedy jeho formát jako *XmlContent*. Název typu resource objektu konfiguračního souboru je *OpenCmsPropertyDialogConfiguration*. Detaily tohoto *XmlContent* dále popisuji v Tab-5.1.

Atribut	Hodnota
Název	OpenCmsPropertyDialogConfiguration
Unikátní název	kiv-property-dialog-configuration
Id	1001

[7] Jména standardních resource typů v OpenCms, které nemají XML schema lze vyčíst z konfiguračního souboru OpenCms *OPENCMS/WEB-INF/config/opencms-vfs.xml* v elementu `<resourcetype>` -> `<type>` z atributu `name`. (př. pro *xmlPage:* `<type class="orgCmsResourceTypeXmlPage" name="xmlpage" id="6"/>`).

Atribut	Hodnota
XML schema URL	/system/modules/cz.zcu.kiv.properties_dialogs/ schemas/property-dialog-configuration.xsd
Explorer view title	Property dialog configuration
Pozice v "New resource" dialogu	New -> structured content

Tab-5.1 – Detailní informace *XmlContent* konfiguračního souboru

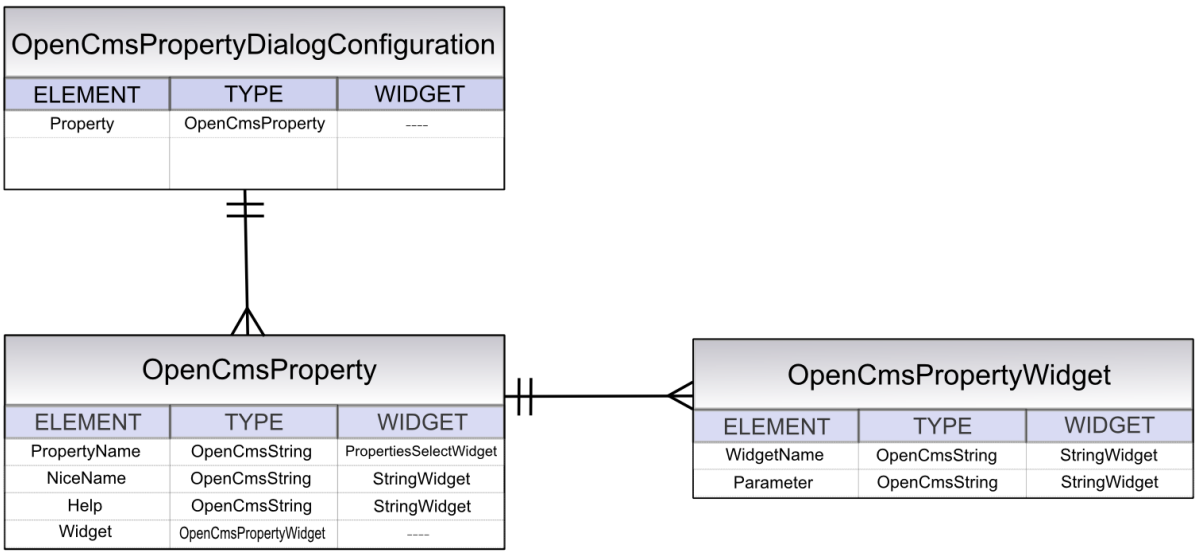
Pro *XmlContent kiv-property-dialog-configuration* je definován i vlastní editor v OpenCms díky nastavení parametrů v rámci modulu[8], který tak pro *XmlContent* představuje kontejner.

Předchozí *XmlContent* resource objekt může obsahovat vnořené uživatelské *XmlContent* resource objekty typu *OpenCmsProperty* a ty následně další vnořené *XmlContent* resource objekty typu *OpenCmsPropertyWidget*, které byly oba vytvořeny v rámci tohoto modulu. Není však pro ně třeba nastavovat detaily, jako v případě prvního *XmlContent* typu, protože je budu využívat pouze jako vnořené struktury a nebudu pomocí nich vytvářet nové resource objekty. Jediné, co zmíním, jsou URL jejich XML schémat:

- *OpenCmsProperty – /system/modules/cz.zcu.kiv.properties_dialogs/schemas/ property.xsd.*
- *OpenCmsPropertyWidget* - /system/modules/cz.zcu.kiv.properties_dialogs/schemas/ property-widget.xsd.

Strukturu a vztah mezi předchozími *XmlContent* typy představuje diagram znázorněný na Obr-5.6.

[8] Toto nastavení se provede v konfiguračním souboru *OPENCMS/WEB-INF/config/opencms-modules.xml* uvnitř elementu, ohraničujícího tento modul a dále uvnitř elementů `<resourcetypes>` a `<explorertypes>`.



Obr-5.6 – Struktura *XmlContent* konfigurace property dialogu.

Widget prvky v property dialogu

Ke každé property položce, která je součástí konfiguračního souboru, lze definovat widget prvek použitý pro její editaci. Seznam těchto widget prvků je v Tab-5.2, kde je také zmíněno, zda bude zobrazení widget prvku v property dialogu ovlivněno zadáním parametru v konfiguračním souboru. Chvíli se pozastavím nad následujícími dvěma widget prvky:

- *Select box with templates* – Více o tomto widget prvku najdete v podkapitole ‘Select box with templates Widget‘.
- *Horizontal line* – Jedná se o vodorovnou čáru, která se samozřejmě nevztahuje k žádné konkrétní property položce. Přesto pokud ji chceme vložit do property dialogu, je nutné nejdříve vytvořit v konfiguračním souboru novou property položku a té přiřadit tento widget prvek. Položku, které widget prvek přiřadíme, můžete beze strachu vybrat libovolně. Na property dialog to nemá žádný vliv.

Widget prvek	Parametr	Popis
Text field	ne	Textové pole.
Select box with parameters	ano	Vysunovací nabídka, jejíž položky jsou zadány v parametru.
Select box with templates	ne	Vysunovací nabídka, jejíž položky jsou templates definované v OpenCms.
Check box	ne	Zatržítko, které ukládá do property položky hodnotu true/false.
Check box group with parameters	ano	Skupina checkbox zatržítek, jejichž popisky a hodnoty jsou definované v parametru.

Widget prvek	Parametr	Popis
Radio group with parameters	ano	Skupina radio zatržítek, jejichž popisky a hodnoty jsou definované v parametru.
VFS URI	ne	Textové pole s možností otevření vyskakovacího okna se souborovým průzkumníkem VFS, které po vybrání souboru vygeneruje do textového pole jeho VFS URI.
Date time	ne	Textové pole s možností otevření vyskakovacího okna s JavaScript kalendářem, které po pohodlném zadání datumu a času vygeneruje tuto zadanou hodnotu do textového pole.
Horizontal Line	ne	Vodorovná čára.

Tab-5.2 – Seznam widget prvků pro property položky.

Seznam widget prvků, ze kterých lze vybírat je možné konfigurovat v XML schématu ke *XmlContent OpenCmsPropertyWidget* v atributu `configuration` elementu `layout` viz Src-5.3. Lokalizovaný properties bundle s řetězci, na které odkazují klíče z `configuration` atributu je součástí modulu a jeho URL ve VFS je */system/modules/cz.zcu.kiv.properties_dialogs/classes/cz/zcu/kiv/properties_dialogs/messages.properties.*

```
<layouts>
  <layout element="WidgetName" widget="SelectorWidget"
       configuration="${key.widget.name.textfield} |
            ${key.widget.name.selectwithparams} |
            ${key.widget.name.selectwithtemplates} |
            ${key.widget.name.checkbox} |
            ${key.widget.name.checkboxgroup} |
            ${key.widget.name.radiogroup} |
            ${key.widget.name.vfs} |
            ${key.widget.name.datetime} |
            ${key.widget.name.horizontalline}" />
</layouts>
```

Src-5.3 – Konfigurace seznamu widget prvků.

Select box with templates Widget

Jedná se o vysunovací nabídku, ve které vybíráme jednu ze šablon definovaných v OpenCms. Jedním z požadavků na tento widget prvek bylo omezení seznamu nabízených šablon vzhledem k aktuální *Site*. Jelikož však v OpenCms nelze jednoduše zjistit, ani přiřadit

šablonu k některé *Site*, bylo zapotřebí implementovat možnost konfigurace. Pro tento účel jsem definoval nový *XmlContent* typ s vlastním editorem. Detaily najdete v Tab-5.3.

Atribut	Hodnota
Název	OpenCmsTemplates
Unikátní název	kiv-template-list
Id	1002
XML schema URL	/system/modules/cz.zcu.kiv.properties_dialogs/ schemas/template-list.xsd
Explorer view title	Seznam šablon pro site
Pozice v "New resource" dialogu	New -> structured content

Tab-5.3 - Detailní informace *XmlContent* pro přiřazení šablon k site

V tomto *XmlContent* typu se uchovává seznam šablon pro *Site*, ve které se resource objekt tohoto typu nachází. Šablona se pro konfigurační soubor vybírá pomocí uživatelského widget prvku *TemplatesSelectWidget*, vytvořeného v rámci tohoto modulu.

TemplateSelectWidget je založena na standardním widget prvku vysunovací nabídky z OpenCms *org.opencms.widgets.CmsSelectWidget*. Položky vysunovací nabídky však oproti tomuto standardnímu widget prvku tvoří seznam všech šablon definovaných v OpenCms[9].

Pro vytvoření resource objektu tohoto typu je třeba dodržet pár následujících pravidel. Resource objekt se musí nacházet v adresáři příslušné *Site* v podadresáři prvního stupně zanoření, jehož jméno je ‘*.config’*. Samotný resource objekt se musí jmenovat *‘template-list.html’*.

Položky pro ‘*Select box widget with templates*‘ se tedy vybírají následovně. Nejprve se ověří existence resource objektu, omezující seznam šablon vzhledem k aktuální *Site*. Ten pokud existuje, tak budou ve widget prvku položky podle něj. Pokud neexistuje, bude ve widget prvku na výběr ze všech šablon v OpenCms.

Ukázku konfiguračního dialogu zobrazuje Obr-A.2 v Příloze A.

Zpracování konfiguračního souboru

Konfigurační soubor je zpracován ve třídě *PropertyDialogHandler* v metodě `List<PropertyBean> getDialogConfiguration(String uri),` kde parametr `uri` udává cestu ke konfiguračnímu souboru ve VFS. Tato metoda zpracuje *XmlContent* konfigurační

[9] Jako template je v OpenCms brán jakýkoliv resource, který je umístěn v podsložce */templates* jakéhokoliv modulu v OpenCms.

soubor tak, že namapuje property položky, které obsahuje, na adekvátní Java Bean. Při vytváření formuláře property dialogu v *PropertyDialogHandler* třídě, kde je volána, pak tato metoda navrací *List* všech property položek z konfiguračního souboru ve formě Java Bean. Součástí modulu jsou dvě Java Bean třídy (viz Obr-5.5):

- *PropertyBean.java* – obsahuje název property položky v OpenCms, titulek, nápovědu a odkaz na *WidgetBean*.
- *WidgetBean.java* – obsahuje název, parametr, *List* popisných textů a *List* hodnot.

V *PropertyBean* atribut `name` představuje název property položky z OpenCms, atribut `niceName` pak titulek, který se zobrazí v property dialogu. Ve *WidgetBean* jsou zpracovány uživatelské parametry. Mimo možnosti získání parametru v podobě, v jaké byl zadán (String), lze navíc získat parametr rozdělený na seznam hodnot a jejich popisků. Druhý případ je možné využít pro následující widget prvky: *Select box with parameters, Check box group with parameters, Radio group with parameters.* Popisky se nabízejí jako volba ve vysunovací nabídce (resp. check box, radio button), hodnoty pak představují skutečnou hodnotu property položky. Aby mohl být parametr rozdělený na tyto dva seznamy, je třeba ho zadat ve formátu jako na Src-5.4. Hodnota a popisek jsou od sebe odděleny znakem '~', jednotlivé parametry jsou odděleny dvojicí znaků '||'.

```
hodnota1 ~ popisek1 || hodnota2 ~ popisek2 || ...
```

Src-5.4 – Formát parametru pro rozdělení na klíče a hodnoty.

5.2.4 Závěr

Ukázku uživatelského property dialogu i s konfiguračními dialogy najdete v Příloze A na obrázkách Obr-A.1 – Obr-A.3.

Největším problémem implementace tohoto modulu bylo zjistit princip výběru property dialog handleru a lokalizovat standardně použité třídy, představující dialog handler v OpenCms. OpenCms nemá implementaci property dialogu řádně zdokumentovanou, tudíž jsem musel řešení hledat přímo ve zdrojových kódech jádra OpenCms.

Také provedení akce uložení hodnoty property položky, změněné přes implementovaný dialog, ukázalo jisté komplikace. HTML kód dialogu je totiž generován uvnitř Java tříd z jádra OpenCms a z nich bylo nutné vyčíst, jakým způsobem identifikovat

formulářové prvky pro property položky, aby systémová akce přiřadila novou hodnotu správnému objektu property položky.

5.2.4.1 Další možná rozšíření modulu

Nakonec zmíním několik dalších funkčností, které by mohli být předmětem dalšího vývoje modulu.

- **Přidat tlačítko odkazující na editaci resource objektu –** Tlačítko by přibylo do základního property dialogu, a kliknutím na něj by se spustila editace resource objektu (pokud by se jednalo o soubor), na němž je property dialog spuštěn.
- **Kontrolovat zděděné property položky –** Pokud by property položka neměla v základním property dialogu příslušného resource objektu nastavenou hodnotu, avšak zdědila by hodnotu od resource objektu předka, pak by se tato hodnota v dialogu zobrazila a současně byla vizuálně odlišena od ostatních.
- **Tlačítko „Basic dialog“ v pokročilém property dialogu –** Pokud se přepnu ze základního property dialogu na pokročilý, již není jiná možnost přímého návratu zpět na základní dialog, než zavřít property dialog a otevřít znovu. Bylo byl tedy dobré modifikovat i pokročilý property dialog a zahrnou do něj odkaz „zpět na základní dialog“.
- **Automatické obnovení dialogu při změně šablony –** Property dialog je mimo jiné vygenerován na základě aktuálně použité šablony resource objektu. Šablonu však mohu změnit i v základním property dialogu, pokud ji konfigurace zahrnuje. Aby se ale změna v sestavení dialogu projevila, je třeba dialog zavřít a znovu otevřít. Lepší by bylo, kdyby se property dialog automaticky obnovil při změně šablony.

5.3 Modul „Report s přístupovými právy“

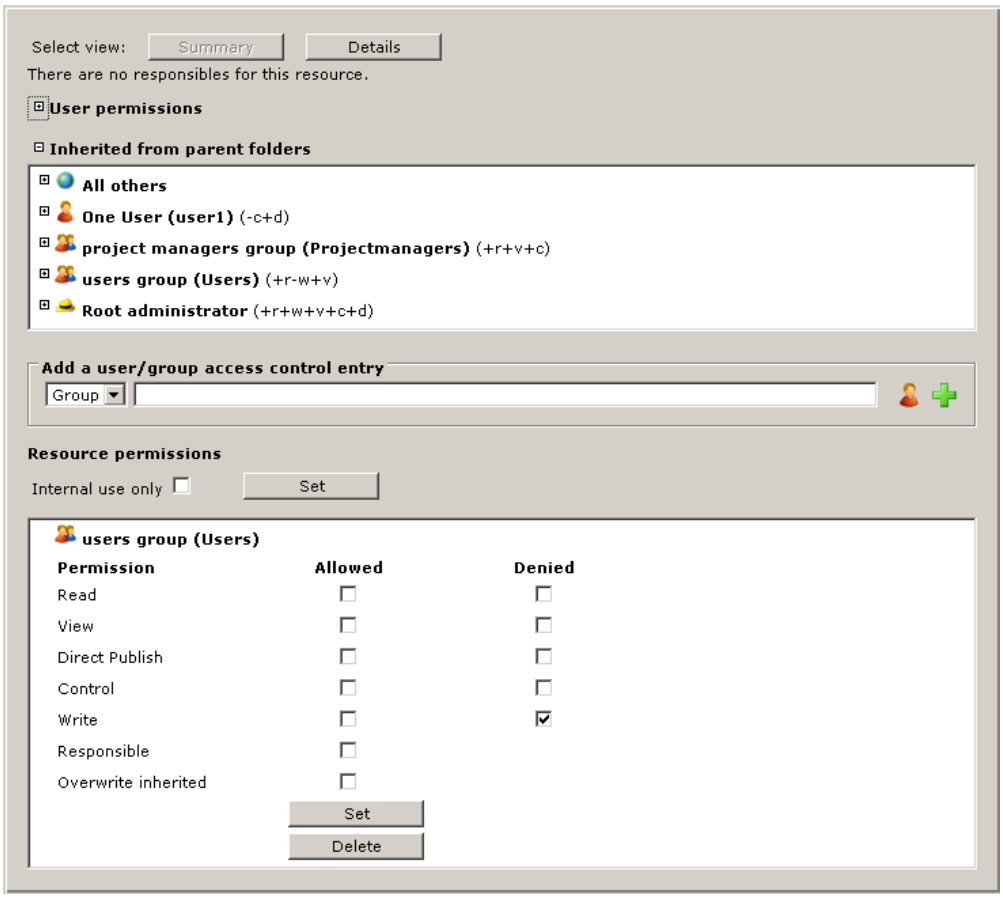
OpenCms disponuje kvalitním systémem přístupových práv, který byl popsán v kapitole 3.4.3. Ten ručí za kontrolovaný přístup k resource objektům ve VFS. Přístupová práva je možné editovat i prohlížet vždy v kontextu s jedním určitým resource objektem. Chybí však možnost zobrazit souhrn přístupových práv vzhledem k vybranému podstromu VFS. Hlavním cílem tohoto modulu je tedy poskytnout možnost vygenerování přehledného reportu sumarizujícího přístupová práva uživatele popř. skupiny právě ve vybraném podstromu VFS v OpenCms workplace.

5.3.1 Standardní zobrazení přístupových práv v OpenCms

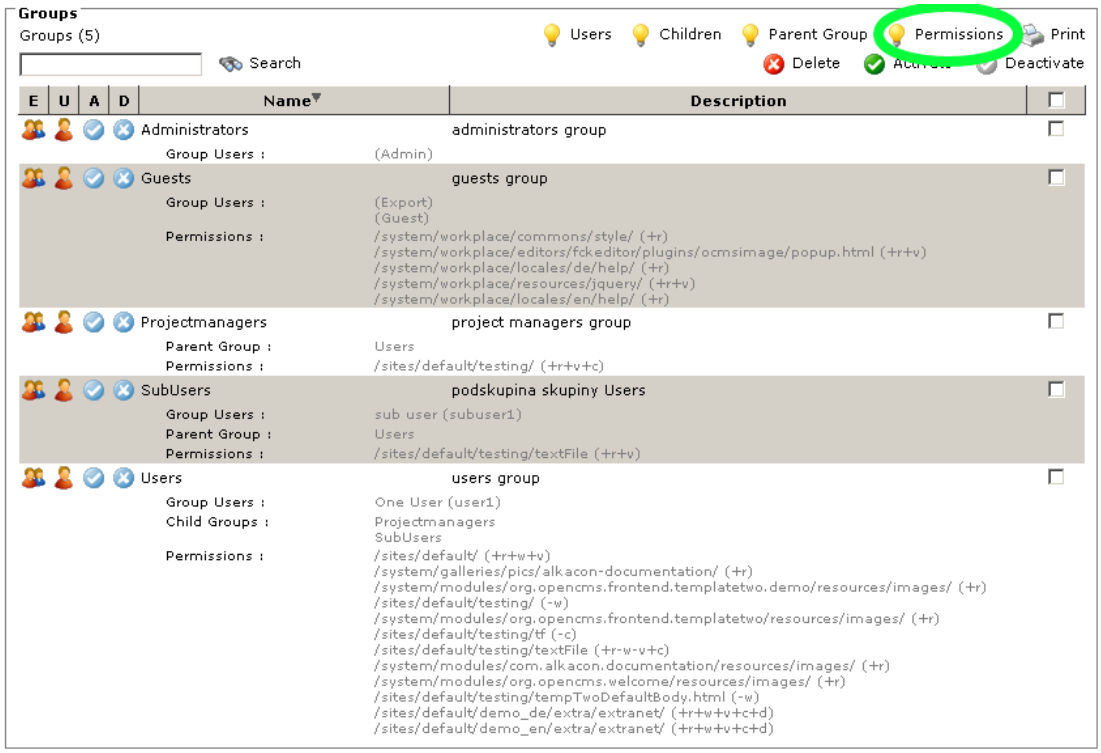
Tento modul se nebude zabývat editací a přiřazováním přístupových práv k resource objektům, ale pouze jejich přehledným výpisem. Jakým způsobem tedy umožňuje zobrazovat přístupová práva standardní instalace OpenCms zmíním v této kapitole.

Možnosti jsou dvě:

- **Ve workplace exploreru –** Přístupová práva lze prohlížet pouze v kontextu s jedním vybraným resource objektem. Přes kontextovou nabídku objektu zvolím možnost *Permissions,* což vyvolá dialog s nastavením přístupových práv znázorněný na Obr-5.7. Vyvolaný dialog reprezentuje JSP soubor ve VFS */system/workplace/commons/chacc.jsp,* který k sestavení dialogu využívá třídu z OpenCms jádra *org.opencms.workplace.commons.CmsChacc.* Takto mohou v OpenCms prohlížet práva všichni uživatelé mající přístup do workplace exploreru a na vybraném resource objektu mají navíc povoleno přístupové právo „view“. Pro editaci práv potřebuje uživatel mít navíc právo „control“ na resource objektu.
- **V administrační části workplace –** V části *‘Account Management -> Group Management’* se nachází seznam skupin definovaných v OpenCms. V tomto seznamu je možné zobrazit pro jednotlivé skupiny resource objekty z celého VFS, na kterých má příslušná skupina nastavená nějaká přístupová práva (ta jsou samozřejmě v seznamu zobrazena také). Implicitně přístupová práva skupin zobrazena nejsou, k tomu je třeba zobrazení v dialogu povolit viz Obr-5.8. Vyvolaný dialog je součástí modulu ze standardní výbavy OpenCms *org.opencms.workplace.tools.accounts*. Reprezentován je JSP souborem */system/workplace/admin/accounts/groups_list.jsp,* který k sestavení dialogu využívá třídu *org.opencms.workplace.tools.accounts. CmsGroupsList* navrženou pro podporu specifického typu OpenCms dialogu se stylem seznamu.



Obr-5.7 – Dialog nastavení přístupových práv na resource objektu.



Obr-5.8 – Dialog se seznamem skupin a jejich přístupových práv na resource objektech

5.3.2 Analýza problému

Dialog z workplace exploreru poskytuje přehledný a detailní pohled na přístupová práva vzhledem k jednomu resource objektu. Vedle nastavení práv k objektu pro aktuálně přihlášeného uživatele zde najdeme i práva různých uživatelů a skupin zděděných z rodičovských adresářů objektu a práva různých uživatelů a skupin definovaná přímo na tomto resource objektu. Navíc dialog disponuje dvěma variantami pohledu – souhrnným a detailním pohledem, ten druhý zobrazuje ještě pár informací navíc. Tento dialog se výborně hodí pro nastavování přístupových práv na resource objektech, avšak přehledný report práv z celého VFS stromu u něj nenajdeme.

Naproti tomu dialog z administrační části sice zobrazuje přístupová práva skupin na resource objektech v celém VFS stromu, ale jedná se pouze o práva přímo nastavená skupinám na resource objektech. Nenajdeme zde práva zděděná od nadřazených skupin, ani práva zděděná od rodičovských resource objektů ve VFS. Zároveň nejsou zahrnuta práva, která se nenastavují přímo pro skupiny, ale nastavují se obecně pro všechny uživatele ve VFS. Také se nedozvíme o právech nastavených pro konkrétní uživatele. Dialog tedy poskytuje částečně funkčnost, která by měla být cílem tohoto modulu, avšak neposkytuje dostatečný detail reportu. Nakonec také dialog není příliš přehledný díky tomu, že resource objekty zmíněné v dialogu jsou prezentovány v neuspořádaném seznamu, ve kterém se nezohledňuje případné zanoření resource objektů ve VFS stromu.

5.3.2.1 Požadavky na rozšíření

Implementovaný modul s rozšířením bude poskytovat uživateli možnost vygenerovat report s přístupovými právy, který bude splňovat následující požadavky:

- Poskytnout dialog pro vygenerování reportu s možností nastavení následujících parametrů:
  - Nastavení kořenového adresáře pro report. Od tohoto adresáře se bude generovat podstrom VFS.
  - Výběr uživatele, nebo skupiny, pro které se bude report v podstromu generovat.
  - Poskytnout možnost výběru typů přístupových práv, které bude report zahrnovat.
- Umožnit snadné spuštění dialogu ve workplace exploreru.
- V reportu bude možné pro vybraného uživatele/skupinu zobrazit u každého resource objektu následující typy přístupových práv:

- Součet přístupových práv principal objektu na resource objektu.
- Práva nastavená přímo resource objektu.
- Práva zděděná z rodičovských adresářů resource objektu.
- Práva zděděná od rodičovských (v případě skupiny), popř. přiřazených (v případě uživatele) skupin.
- Práva nastavená pro systémovou roli „All others“.

5.3.2.2 Výběr vhodného dialogu pro report

Před samotnou implementací bylo nutné rozhodnout, jaký typ OpenCms dialogu pro report použít a kde zpřístupnit uživatelům odkaz k jeho otevření.

OpenCms poskytuje několik tříd, implementujících dialogy ve workplace. Třídy poskytují metody, které umožní zachovat pro dialogy konzistentní OpenCms vzhled. Tyto dialogy mohou být využity jak ve workplace exploreru, tak v administrační části workplace. Lze vybírat ze široké palety dialogů, z nichž zmíním alespoň některé:

- *org.opencms.workplace.CmsMultiDialog* – třída umožňuje vytvořit dialog schopný vykonávat operace nad více než jedním VFS resource objektem naráz. Jedná o operace jako kopírování, mazání, „touch“ apod.
- *CmsTabDialog* – třída umožňující vytvořit dialog s tabulkovým stylem. Příklad konkrétního tabulkového dialogu je *CmsPropertyAdvanced* dialog, o kterém byla zmínka v kapitole 5.2.
- *A_CmsListDialog* – Abstraktní třída poskytující metody pro vytvoření dialogu, který zahrnuje widget prvek typu seznamu.
- Dále existuje v OpenCms několik dalších konkrétních dialogů, které jsou využity pro jednu specifickou operaci. Detailní informace k těmto dialogům lze nalézt v Javadoc k OpenCms [12].

Všechny tyto dialogy mají jednoho společného předka, od kterého jsou odvozeny. Tím předkem je třída *org.opencms.workplace.CmsDialog* a ta již poskytuje mnoho metod využitelných obecně pro různé typy dialogů. Pokud tedy chci vytvořit dialog s konzistentním OpenCms vzhledem, musím svou dialog třídu odvodit minimálně od třídy *CmsDialog*.

Jak by měl dialog pro report přístupových práv vypadat?

Pro splnění požadavků na report budu potřebovat dialog složit ze tří částí.

1. Blok s formulářovými prvky pro nastavení reportu.
2. Blok pro samotný výpis reportu.
3. Blok pro tlačítkovou lištu, kde bude tlačítko pro uzavření dialogu.

První blok dialogu je dost specifický, jelikož obsahuje sadu formulářových prvků šitou přímo na vykreslení reportu. Nevyužiji tedy k jeho sestavení žádný již existující specifický dialog, bude stačit vymezit tento blok pomocí metod *dialogBlockStart()* a *dialogBlockEnd()* z *CmsDialog*. Pro druhý blok bude stačit vytvořit oblast, kam se bude generovat text s reportem. K tomu se nabízí metody *dialogWhiteBoxStart()* a *dialogWhiteBoxEnd()* opět ze třídy *CmsDialog.* Tyto metody vymezují rámeček s bílým pozadím, na který je možné vypisovat text. Jelikož *CmsDialog* poskytuje několik metod i pro vytvoření různé sady tlačítek, využiju tuto třídu i pro sestavení třetího bloku. Konkrétně mi pro vytvoření tlačítka pro zavření dialogu bude stačit metoda *dialogButtonsClose().*

Z předchozího odstavce vyplývá, že třídu, implementující dialog pro report přístupových práv, odvodím od základní třídy pro vytváření dialogů v OpenCms - třídy *CmsDialog.*

Kam ve workplace umístit odkaz pro otevření dialogu?

Aby byl dialog s reportem ve workplace snadno přístupný a měl stálou pozici, není vhodné jej spojovat s jedním konkrétním souborem ve VFS na frontendu. Potom se tedy nabízí následující možnosti umístění odkazu:

1. Do kontextového menu resource objektu.
2. Do tlačítkové lišty ve workplace exploreru.
3. Do administrační části workplace.

První možnost se mi v porovnání s ostatními zdála nejméně vhodnou již od začátku. Sice by odkaz nebyl spojen s jedním konkrétním resource objektem (odkaz by se nacházel v kontextovém menu jakéhokoli resource objektu), ale přece jen umístění by bylo vzhledem k požadavku na generování reportu z celého podstromu poněkud nelogické. Tato možnost je podobná s druhou možností a vyvolání kontextového menu představuje vzhledem k ní ještě jedno kliknutí (vyvolání kontextové nabídky) navíc.

Druhá možnost má největší výhodu v umístění odkazu. Ten se nachází ve workplace exploreru, ve kterém se všichni OpenCms uživatelé pohybují nejčastěji. V exploreru je navíc

součástí stále viditelné lišty tlačítek a tudíž přístupný z „první ruky“. Nevýhoda této možnosti (oproti třetí možnosti)je pak nutný zásah do jádra OpenCms (první možnost vyžaduje taktéž nutný zásah do jádra). V praxi by pak mohl nastat následující problém. Pokud by se modul instaloval do OpenCms systému, ve kterém již byla tlačítková lišta modifikována, byla by ta původní přepsána tlačítkovou lištou z tohoto modulu, takže by se původní modifikace ztratila.

Třetí možnost je vytvoření nové položky v administrační části workplace, ve které se uživatelé nepohybují tak často jako v exploreru. Na druhou stranu OpenCms umožňuje snadnou rozšiřitelnost administrační části, bez nutného zásahu do souborů jádra OpenCms.

Zbývá tedy rozhodnout mezi druhou a třetí možností. Jelikož vygenerování reportu nebude zrovna běžnou akcí, které by bylo potřeba využívat příliš často, umístění v administrační části přílišným omezením není. Naopak možnost nezasahovat do souborů jádra přináší ve finále větší užitek. Odkaz pro otevření dialogu tedy v rámci modulu vytvořím jako položku administrační části. Modifikovaný soubor s tlačítkovou lištou s modulem spojovat nebudu, tak aby se při importu modulu nepřepsal původní soubor jádra. Přesto ale poskytnu v kapitole 5.3.3.3 návod, jak si tlačítko do tlačítkové lišty přidat manuálně.

5.3.2.3 Detailní pohled na přístupová práva vzhledem k resource objektu a principal objektu

V této kapitole nebudu popisovat systém přístupových práv v OpenCms. Ten byl již popsán v kapitole 3.4.3. Zaměřím se spíše na popis tříd z OpenCms jádra a jejich metod, které umožňují provádět různé operace spojené s přístupovými právy.

Vstupními parametry dialogu, generujícího report, bude VFS strom vygenerovaný na základě poskytnuté URI kořenového adresáře a principal objekt, pro kterého chci zjistit oprávnění na resource objektu. Jak bylo řečeno v kapitole 3.4.3.2, tyto parametry zaobaluje třída *CmsAccessControlEntry* (ACE). Cílem bude získat objekt *org.opencms.security. CmsPermissionSet*, který obsahuje sadu přístupových práv (allowed i denied) a metody pro výpis těchto práv. Tento objekt budu potřebovat získat ve spojitosti právě s resource objektem a zároveň principal objektem.

Výchozím bodem je třída *CmsObject*, která poskytuje následující, z hlediska tohoto modulu důležité metody:

- `List getAccessControlEntries(java.lang.String resourceName, boolean getInherited)` – vrací seznam ACE položek daného resource objektu.

- `CmsAccessControlList getAccessControlList(java.lang.String resourceName, boolean inheritedOnly)` - vrací *CmsAccessControlList* (ACL), tedy sumarizované ACE položky vzhledem k danému resource objektu. ACL se bude hodit zvláště pro získání součtu přístupových práv, které jsou jedním z požadavků modulu.

Obě metody také zahrnují parametr, zda do ACE položek zahrnout i zděděné položky z rodičovských resource objektů, což je přímo také jedním z požadavků.

Nyní tedy již budu schopen s pomocí předchozích metod získat všechny ACE položky daného resource objektu. Zbývá najít filtr, kterým získám položky pouze vybraného principal objektu. Pro tento účel se nabízí následující metody tříd *CmsAccessControlEntry:*

- `CmsUUID getPrincipal()` – vrátí principal objekt (ve skutečnosti jeho unikátní identifikátor) přiřazený ACE položce.
- `CmsPermissionSet getPermissions()` – vrátí požadovanou sadu přístupových práv.
- `void setFlagsForPrincipal(I_CmsPrincipal principal)` – nastaví příznak pro identifikaci typu principal objektu spojeného s ACE položkou. Tuto metodu využiji při získávání práv pro systémovou roli „All others“.

a *CmsAccessControlList*:

- `CmsPermissionSetCustom getPermissions(CmsUser user, java.util.List groups, java.util.List roles)` – vrátí součet přístupových práv uživatele, jeho skupin a jeho rolí. Pokud navíc ACL zahrnuje i zděděné ACE položky od rodičovských tříd, získám touto metodou celkový součet práv uživatele, který je jedním z požadavků.
- `CmsPermissionSetCustom getPermissions(CmsUUID principalId)` – vrátí součet přístupových práv *CmsPrincipal* objektu. Tuto metodu zase využiji pro získání součtu práv skupiny.

Metodami zmíněnými v této kapitole již dokážu pokrýt všechny požadavky, týkající se typů přístupových práv principal objektu na resource objektu.

5.3.3 Řešení problému

V této kapitole se budu nejprve věnovat opět struktuře modulu. Následovat bude popis způsobu generování VFS stromu resource objektů, u kterých budou zjišťována přístupová práva. Zmínka padne i o vytvoření položky v administrační části, která bude odkazovat na dialog reportu.

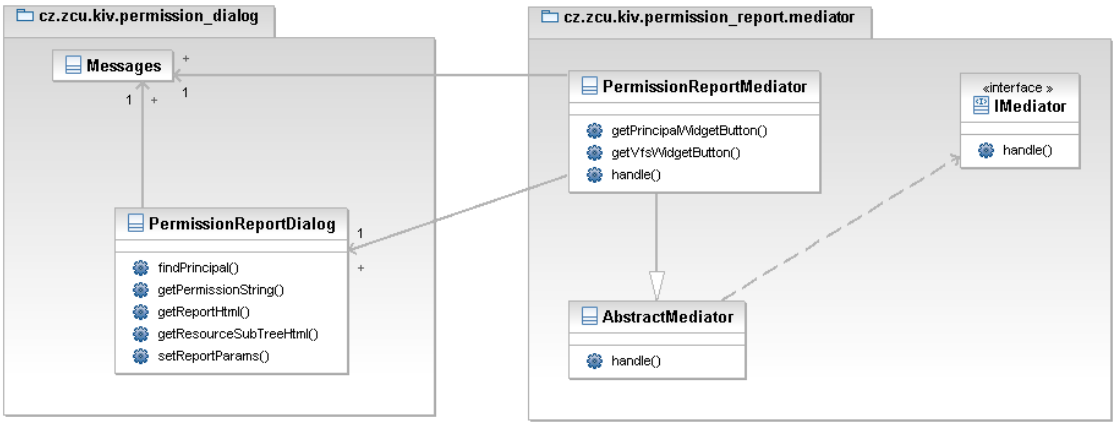
5.3.3.1 Struktura modulu

Název modulu je *cz.zcu.kiv.permission_report*. V něm jsou zahrnuty následující adresáře:

- Systémové adresáře **classes** a **lib** popsané v kapitole 3.5.1.
- adresáře **elements, resources** popsané v kapitole 3.5.1.
- **dialogs** – Zahrnuje JSP s dialogem generujícím report přístupových práv.

  Dále jsou k modulu připojeny jako externí resource objekty následující adresáře:
- */system/workplace/admin/permissionreport/* - adresář reprezentující položku administrační části. Obsahuje pouze JSP soubor, který si začleňuje JSP soubor s dialogem z modulu.
- */system/workplace/resources/tools/permissionreport/* - adresář zahrnuje doplňující soubory pro administrační položku. Např. ikony.

Modul disponuje také vlastní JAR knihovnou tříd *cz.zcu.kiv.permission_report* v adresáři */lib*. Ta zahrnuje třídy implementující aplikační vrstvu modulu. Následuje popis JAR knihovny. Diagram tříd knihovny znázorňuje Obr-5.9.



Obr-5.9 – Diagram tříd z knihovny modulu *cz.zcu.kiv.permission_report.jar*

Package cz.zcu.kiv.permission_report

Základní balík modulu zahrnuje třídu *PermissionReportDialog.java*, která implementuje samotný dialog reportu. Obstarává prakticky veškerou důležitou funkčnost modulu. Poskytuje pomocné funkce pro vygenerování celého dialogu. Zpracovává parametry dialogu, na jejichž základě vytváří report s přístupovými právy (generuje VFS podstrom i sady požadovaných typů přístupových práv).

Dále se v balíku nachází opět třída *Messages.java* poskytující přístup k lokalizovaným řetězcům modulu viz kapitola 5.1.2.

Package cz.zcu.kiv.permission_report.mediator

Opět jako v prvním implementovaném modulu, i zde tento balík obsahuje třídy, které jsou součástí „Mediator pattern“ architektury.

Jediným prostředníkem mezi JSP dialogem a hlavní třídou *PermissionReportDialog* v tomto modulu je třída *PermissionReportMediator.java*. Jejím úkolem je předávat mezi dialogem a hlavní třídou modulu parametry ve formě atributů implicitních objektů. S pomocí těchto parametrů, přicházejících z JSP, vytvoří hlavní třída požadované HTML pro dialog. JSP dialog pak využívá JSTL pro čtení parametrů a vyhne se tak nepřehledným skriptletům.

5.3.3.2 Generování VFS stromu s resource objekty

Pro generování VFS stromu s resource objekty od objektu daného parametrem dialogu se ve třídě *PermissionReportDialog* používají dvě metody:

- `String getReportHtml(String resourceName, String principalName, boolean sum, boolean own, boolean folderin, boolean groupin, boolean all)` – generuje HTML pouze pro resource objekty, které jsou přímými potomky adresáře daného v parametru (`resourceName`). Další parametry jsou využity pro generování různých typů (boolean parametry metody určují, zda se bude generovat příslušný typ práv) přístupových práv daného principal objektu (`principalName`) na konkrétním resource objektu.
- `String getResourceSubTreeHtml(String resourceName, CmsPrincipal principal)` – Tato metoda je volána z předchozí metody a rekurzivně generuje HTML pro zbytek VFS podstromu od resource objektu daného v parametru (to je jeden z přímých potomků kořenového objektu z předchozí metody).

Obě metody také pro aktuálně generovaný resource objekt sestavují zároveň i HTML s informacemi o přístupových právech a to pomocí metody:

- `String getPermissionString(CmsResource resource, CmsPrincipal principal).`

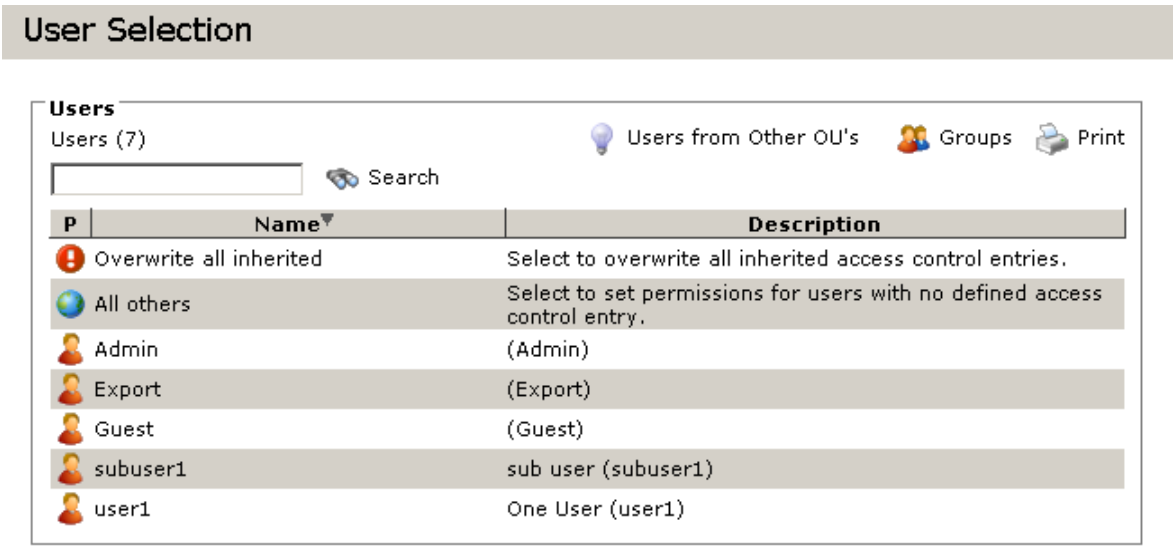
Parametry pro generování reportu

Parametry pro report je možné zadávat v horní části dialogu, obsahujícího report viz Příloha A, Obr-A.4. Souhrn parametrů zobrazuje Tab-5.4.

Parametr	Popis	Widget prvek
Report root folder	Kořenový adresář, od kterého se bude generovat VFS podstrom resource objektů, u kterých se budou zkoumat přístupová práva.	Textové pole (zadání možné pomocí widget prvku *VfsFileWidget*)
Principal	Uživatel, nebo skupina uživatelů, pro kterého(kterou) se budou zkoumat přístupová práva na konkrétním resource objektu.	Textové pole (zadání možné pomocí widget prvku *CmsPrincipalWidget)*
Show permissions summary (SUM)	Volba, zda zobrazit součet přístupových práv principal objektu na resource objektu.	check box
Show own entries (OWN)	Volba, zda zobrazit přístupová práva pro principal objekt nastavená přímo na příslušném resource objektu.	check box
Show inherited from folder (FOL)	Volba zda zobrazit přístupová práva zděděná z rodičovských adresářů resource objektu.	check box
Show inherited from groups (GRP)	Volba, zda zobrazit přístupová práva zděděná od rodičovské skupiny, popř. přiřazených skupin.	check box
Show all others (ALL)	Volba, zda zobrazit přístupová práva pro systémovou roli "All others".	check box

Tab-5.4 – Parametry dialogu pro generování reportu s přístupovými právy

VfsFilewidget otevře v dialogu vyskakovací okno souborového průzkumníka ve VFS, které po vybrání souboru vygeneruje do textového pole jeho VFS URI. *CmsPrincipalWidget* otevře vyskakovací okno s výběrem uživatele, či skupiny ze seznamu definovaných v OpenCms viz Obr-5.10.



Obr-5.10 – *CmsPrincipalWidget* pro výběr uživatele, či skupiny

Formát HTML generovaného VFS stromu

Metody generují HTML reportu ve formě neuspořádaného seznamu (`ul`), jehož položky (`li`) tvoří jednotlivé resource objekty z VFS stromu. Jelikož se předpokládá, že VFS podstrom může být poměrně rozsáhlý, je potřeba neuspořádaný seznam vhodným způsobem formátovat.

K tomuto účelu jsem využil plugin *Treeview* do JavaScript knihovny *jQuery*[10]. *Treeview* umožňuje flexibilní transformaci neuspořádaného seznamu na rozkládací a zasunovací strom. Více informací o tomto pluginu, včetně demo ukázek a odkazu na dokumentaci, je dostupných na webových stránkách pluginu [13]. Příklad vygenerovaného reportu, tak jak ho naformátoval *Treeview* si prohlédněte v Příloze A, na Obr-A.5.

5.3.3.3 Vytvoření položky administrační části

Pro zobrazení dialogu s reportem přístupových práv jsem přidal do administrační části workplace položku, která na dialog odkazuje. Odkaz se zobrazí jako ikona přímo v hlavní části administračního pohledu, jak je vidět na Obr-5.11.

[10] Knihovna *jQuery v 1.2.3* je součástí standardní instalace OpenCms 7.0.5

Obr-5.11 – Administration view v Opencms workplace

OpenCms poskytuje možnost rozšíření administrační části, takže přidání položky je již jednoduchým, přímočarým úkolem.

Všechny položky administrační části jsou načítány z adresářů umístěných ve VFS pod následujícím URI:

- */system/workplace/admin/*

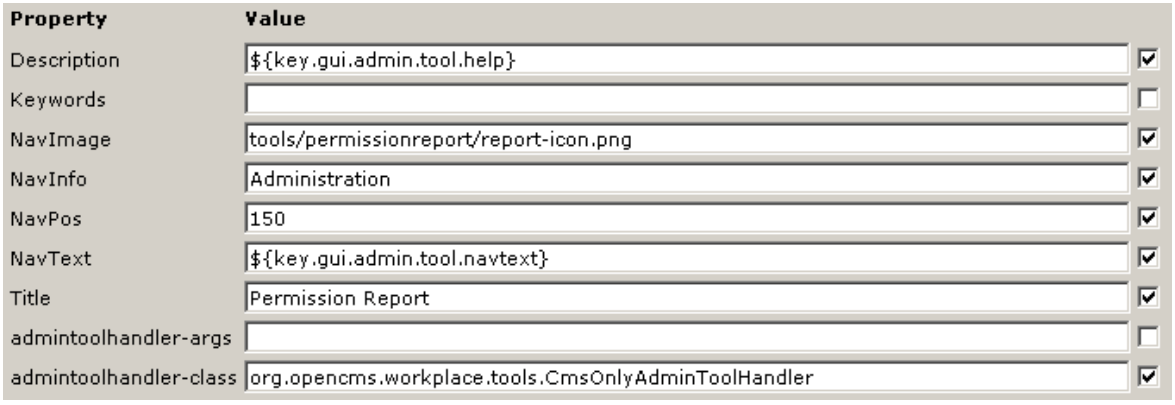
OpenCms při načítání workplace vytvoří ikonu pro každý adresář na předešlé poloze na základě hodnot property položek zadaných na resource objektu složky. Důležité je také zmínit, že adresáře a jejich property položky jsou načítány pro administrační část pouze z Online projektu, takže jakékoli změny je třeba nejprve publikovat, aby se projevily.

V rámci tohoto rozšíření jsem tedy vytvořil administrační položku */system/workplace/admin/permissionreport/*, na které jsem nastavil property položky dle Obr-5.12. Použité lokalizační hodnoty jsou umístěné v modulu reportu v Java properties souboru */system/modules/cz.zcu.kiv.permission_report/classes/cz/zcu/kiv/permission_report/*. Hodnota property položky *NavImage* určuje umístění souboru s ikonou pro administrační bod. Uvedená cesta je relativní vzhledem k povinnému umístění ikon v OpenCms, takže výsledná URI souboru ikony je */system/workplace/resources/tools/permissionreport/report-icon.png.*

Adresář nové administrační položky obsahuje pouze soubor *index.jsp,* který volá pouze akci *include,* vkládající soubor *permission-report-dialog.jsp* z modulu rozšíření.

Aby byla nová položka administrační části součástí modulu rozšíření, připojil jsem adresář s položkou administrační části a adresář obsahující ikonu této položky k modulu jako externí resource objekt[11].

[11] Připojení externích resource objektů k modulu se provede v Administrační části *Module Management -> Editace příslušného modulu -> Module Resources.*



Obr-5.12 – Property položky na adresáři nového administračního bodu.

Alternativa – umístění odkazu do tlačítkové lišty workplace exploreru

Většinová část workplace explorer okna je sestavena v souboru */system/workplace/views/explorer/explorer_files.jsp*, který k sestavení využívá třídu *org.opencms.workplace.explorer.CmsExplorer*. Součástí předchozího JSP souboru je i tlačítková lišta, do které budeme vkládat nové tlačítko. Ta je vytvořena v JavaScript souboru */system/workplace/resources/commons/explorer.js* ve funkci `function displayHead(doc, pages, actpage)`.

Pro přidání tlačítka do lišty je třeba nejprve deklarovat proměnnou tlačítka (viz Src-5.5).

```
// tlacitko pro zobrazeni permission reportu
var btPermissionReport = "";
```

Src-5.5 – deklarování proměnné tlačítka.

Následně vytvoříme tlačítko (viz Src-5.6) pomocí další funkce ze stejného JavaScript souboru `function button(href, target, image, label, type)`. Parametry funkce jsou:

- **href –** URL souboru, na které tlačítko odkazuje,
- **target –** cíl umístění tlačítka,
- **image –** URI ikony tlačítka,
- **label –** popisek tlačítka,
- **type –** typ tlačítka (1-ikona a text, 2-pouze text, 0-default-pouze ikona).

```
//vytvoreni permission button
btPermissionReport = button(vr.servpath +
      "/system/modules/cz.zcu.kiv.permission_report/dialogs/permission-
      report-dialog.jsp" ,"explorer_files", "principal.png", 'Permission
      Report', 1);
```

Src-5.6 – Vytvoření tlačítka.

Nakonec tlačítko přidáme do lišty (viz Src-5.7) a máme hotovo.

```
...
+ "<tr>\n"
+ buttonSep(0, 0, 0)
+ button("javascript:top.histGoBack();", null, "back.png", vr.langback,
      buttonType)
+ btUpload
+ btSearch
+ btWizard
+ btUp
+ btPermissionReport

+ buttonSep(5, 5, 1)
...
```

Src-5.7 – část tlačítkové lišty s přidaným tlačítkem

5.3.4 Závěr

Ukázku reportu s přístupovými právy najdete v Příloze A na obrázkách Obr-A.4 a Obr-A.5.

Největším problémem implementace tohoto modulu bylo bezpochyby nastudování systému přístupových práv v OpenCms, který není zrovna triviální a také OpenCms core API, které má u tříd, týkajících se přístupových práv rezervy v dokumentaci. Funkčnost některých metod tedy bylo nutné vyčíst přímo ze zdrojových kódů.

Další komplikace nastaly při pokusu o přidání tlačítka do workplace exploreru. Workplace explorer je složen z HTML rámců, které jsou sestavovány v JSP souborech, které navíc k sestavení používají různé Java třídy. A jelikož je explorer velká JavaScript aplikace, v těchto třídách jsou ještě schované odkazy na tyto JavaScript soubory, implementující jeho funkčnost. Velkým problémem se tedy stala nejprve lokalizace těchto JavaScript souborů a

jejich funkcí, ve kterých bylo nutné provést změny a následně zjištění principu fungování této aplikace.

Nečekaný problém se také objevil při generování reportu při současné pozici uvnitř některé ze *Site* položek ve VFS. OpenCms totiž rozlišuje ve VFS mimo jiné kořenovou cestu k resource objektu (dále *rootUri*) a cestu k objektu vzhledem k aktuální *Site* položce (dále *siteUri*). Pro report je potřeba zadávat parametr kořenového resource objektu pomocí *rootUri*, na druhé straně funkce, pomocí kterých vyhledávám resource objekty ve VFS podstromu uvažují *siteUri* (alternativní funkce k těmto pak OpenCms API nenabízí). Výsledkem toho je, že pokud se nacházím uvnitř některé *Site* položky, tak se resource objekty z podstromu vyhledávají pomocí *rootPath*, která se od *sitePath* v tu chvíli liší a tudíž není nalezen žádný resource objekt. Po nějakém čase stráveném při hledání řešení v OpenCms API, se mi podařilo problém vyřešit. Třída *org.opencms.file.CmsRequestContext*, která nese informace o OpenCms kontextu aktuálně přihlášeného uživatele, totiž dovoluje pomocí metody `setSiteRoot(java.lang.String root)` změnit aktuální *siteUri*. Těsně před generování VFS stromu do reportu tedy změním *siteRoot* na default hodnotu (tou je prázdný řetězec) a tím zajistím, že *siteUri* a *rootUri* budou shodné. Po dokončení generování pak nastavím *siteRoot* na původní hodnotu a je hotovo.

5.3.4.1 Další možná rozšíření modulu

Nakonec zmíním několik dalších funkčností, které by mohly být předmětem dalšího vývoje modulu.

- **Lazy-load renderování VFS stromu s resource objekty** – jQuery plugin *Treeview* ve verzi 1.4 umožňuje renderovat pouze neuspořádaný seznam s kompletně načtenými položkami (resource objekty). Do verze *Treeview 1.5* je však v plánu implementovat funkčnost lazy-load renderování stromu takovým způsobem, že budou vyrenderovány pouze aktuálně požadované elementy stromu. Této funkčnosti by se mohlo využít i při vytváření seznamu z VFS podstromu, který může být poměrně rozsáhlý a v současné době se vždy musí sestavit celý.
- **Propojení reportu s dialogem pro editaci přístupových práv** – U každého resource objektu by mohl být odkaz na tento OpenCms dialog.
- **Animovaný indikátor průběhu generování reportu** – Pokud report generuji s pomocí dialogu, který není v Administrační části, tak v případě rozsáhlejšího VFS

podstromu akce může trvat v řádu desítek vteřin. Bylo by tedy vhodné doplnit dialog o nějaký JavaScript indikátor ukazující, že report se opravdu generuje.

- **Propagovat nastavení přístupových práv na rodičovské adresáře až ke kořenu podstromu** – Pokud bude nastaven určitý typ přístupových práv na resource objektu v nižších úrovních podstromu VFS, pak bude v reportu u jeho rodičovských resource objektů tato změna indikována (může být barevně odlišena od ostatních přístupových práv v reportu).

5.4 Nasazení modulů této kapitoly

Moduly byly vyvíjeny a testovány jako open-source projekty pod licencí LGPL za použití následujícího software a příslušných verzí:

- Windows Vista Business SP1 CZ 32bit,
- Java JDK 1.6.0_13, Java HotSpot Client VM build 11.3-b02,
- Apache Tomcat Version 6.0.18,
- MySQL 5.0.45-community-nt,
- OpenCms 7.0.5,
- Eclipse Platform Version 3.4.1.

Domovské stránky pro oba moduly poskytuje server SourceForge.net. Na stránkách je možné moduly stáhnout a získat o nich detailní informace. Stránky jsou:

- http://sourceforge.net/projects/opencmspropsext/ - pro modul „Parametrizovaný properties dialog“.
- http://sourceforge.net/projects/opencmsext/ - pro modul „Report s přístupovými právy“.

Oba moduly byly testovány průběžně během implementace na instalaci OpenCms, na které byly vyvíjeny. Zároveň finální verze modulů v rámci této práce byly nasazeny na jinou instalaci OpenCms (avšak stejnou verzi 7.0.5), než vývojovou. Testování vykazovalo stejné výsledky na obou verzích instalace a ty odpovídaly předpokladům. Nebylo možné moduly otestovat podle původního plánu na OpenCms, běžícím v ostrém (www.kiv.zcu.cz) popř. beta (beta.kiv.zcu.cz) provozu na KIV. Důvodem je, že na těchto referenčních strojích se dosud neuskutečnil přechod ze starší verze OpenCms 6.2.3 na novou verzi OpenCms 7.0.5. Nově

vyvíjené moduly jsou s verzí OpenCms 6.2.3 nekompatibilní. Pro jejich nasazení na starší verzi by bylo nutné provést pár změn, které však již nebudou na moduly aplikovány kvůli brzy plánovanému přechodu referenčních strojů na verzi OpenCms 7.0.5.

6 Implementace modulů pro web katedry KIV

Zatímco předchozí kapitola se týkala rozšíření jádra OpenCms, tato kapitola se bude zabývat popisem modulů pro webové stránky katedry KIV.

Web katedry KIV prochází již od listopadu roku 2004 transformací na systém OpenCms. V rámci této transformace bylo nejprve předěláno jádro webových stránek a dosud stále probíhá přidávání doplňujících funkčností do systému ve formě dynamických modulů. V současné době jsou nasazeny (v „ostrém“ popř. testovacím provozu) následující moduly: *Aktuality, Osoby, Témata projektů, Login, Produkty.*

Ve vývoji jsou moduly: *Skupiny a Granty, Publikace.*

Další plánované moduly pak jsou: *Akce, Vyhlášky a pokyny, Zápisy z vedení, Techreporty, Kalendář a termínovník, Obhájené práce.*

Součástí této práce je implementace dvou následujících modulů:

- **Modul Osoby –** Evidence pracovníků a doktorandů katedry KIV, včetně jejich aktivit a výuky.
- **Modul Produkty –** Evidence software na katedře spolu s licencí.

6.1 Nezbytné předpoklady modulů

V této kapitole zmíním základní modul, který je nutným předpokladem používání doplňkových modulů pro web katedry.

6.1.1 Modul Common

Většina doplňujících modulů pro web katedry je závislá na základním modulu, jehož název je *Common*, celé jméno balíku modulu pak *cz.zcu.kiv.common*. Tento modul zaobaluje funkčnosti, které jsou společné pro více modulů. Jeho hlavní část představuje knihovna *webkiv-db* s implementací DAO vrstvy vystavěné nad Spring JDBC.

6.1.1.1 Knihovna webkiv-db

Knihovnu navrhnul a její základ implementoval Radek Muzika na začátku roku 2008. Jedná se o knihovnu, která zajišťuje přístup do databáze na KIV, kde jsou skladována data pro web katedry. Jádro knihovny je postaveno nad frameworkem Spring, ze kterého jsou využity funkce IOC kontejneru. Pro přístup do databáze je pak využita knihovna Spring JDBC, která je součástí frameworku. Podrobné informace i s návodem použití k této knihovně jsou k dispozici na [19].

6.2 Architektura modulů

V této kapitole popíši společnou strukturu obou modulů a architekturu jejich aplikační vrstvy.

6.2.1 Struktura modulů

Moduly této kapitoly opět dodržují doporučenou strukturu modulů, popsanou v kapitole 3.5.1, stejně jako moduly v kapitole 5. Obsahují tedy složky *classes*, *elements*, *lib*, *resources* a navíc ještě složku *pages*, který obsahuje hlavní funkční JSP stránky modulů.

6.2.2 Mediator pattern architektura

Mediator pattern architektura v této části je první verzí stejné architektury[1], která je popsána v kapitole 5.1.1 a tak obsahuje ještě některé nedostatky, které byly následující verzí odstraněny. Z toho důvodu bude popis architektury velice stručný, protože v současné době se již pracuje na transformaci modulů na novou verzi Mediator pattern architektury.

V architektuře jsou zahrnuty kromě Mediator tříd ještě další specializované třídy, které se nazývají *Form Handler* třídy. Oba typy tříd v modulech pro katedru působí jako prostředníky mezi JSP stránkami modulů a objekty datové vrstvy.

6.2.2.1 Mediator

Java třídy typu Mediator umožňují aplikaci přístup k datové vrstvě. Využívají pouze metody „*Read*“ žádaného DAO objektu. Z JSP stránky si pak snadno může aplikace prostřednictvím Mediator třídy volat jednotlivé metody z DAO.

[1] Oba dva moduly byly implementovány ještě před vznikem druhé verze Mediator pattern architektury, na které jsou již postavené moduly z kapitoly 5.

6.2.2.2 Form handler

Tyto třídy jsou ještě specializovanějším typem Mediator tříd. Navíc od nich využívají z příslušných DAO objektů zbytek základních persistenčních CRUD metod – „*Create*“, „*Update*“, „*Delete*“.

6.3 Modul Osoby

Modul Osoby doplňuje web katedry o funkčnosti spojené s evidencí pracovníků a doktorandů katedry. Modul by měl transformovat funkčnost stejnojmenného modulu z původní verze webu katedry, která ještě nebyla založená na OpenCms a navíc ji rozšířit o další vlastnosti.

Modul byl implementován jako jeden z prvních doplňujících OpenCms modulů webu katedry, protože jeho funkčnost je podmíněna vytvořením DAO vrstvy pro osoby katedry, na které bude dále záviset většina ostatních modulů.

6.3.1 Požadavky na modul

Modul Osoby bude poskytovat návštěvníkům webu katedry přehledný seznam zaměstnanců s možností zobrazit detailní informace o každém z nich. Detail pak bude zahrnovat základní informace o osobách, jejich aktivity a členství v různých organizacích.

U modulu osob je jedním z požadavků umožnit vice náhledů na pracovníky. Mělo by tedy být možné řadit je podle vice kritérií. Nový systém by měl rovněž umožňovat přidat pozice vedoucích jednotlivých podoborů katedry (softwarové inženýrství, grafika…), tajemníků katedry atd.

Dále pak bude modul splňovat následující požadavky:

- Seznam osob bude možné filtrovat podle různých kategorií. V každé kategorii pak budou osoby seřazené podle pozic, které mají přiřazeny.
- U osob budou evidovány jejich aktivity a členství v různých organizacích.
- U osob, které jsou doktorandy na katedře, bude možné evidovat jejich školitele. Na druhé straně, u školitelů bude možné zobrazit seznam jejich doktorandů.
- V prezentační vrstvě poskytnout osobám s příslušnými přístupovými právy možnost provádět základní *CRUD* operace nad jednotlivými osobami a dále kategoriemi,

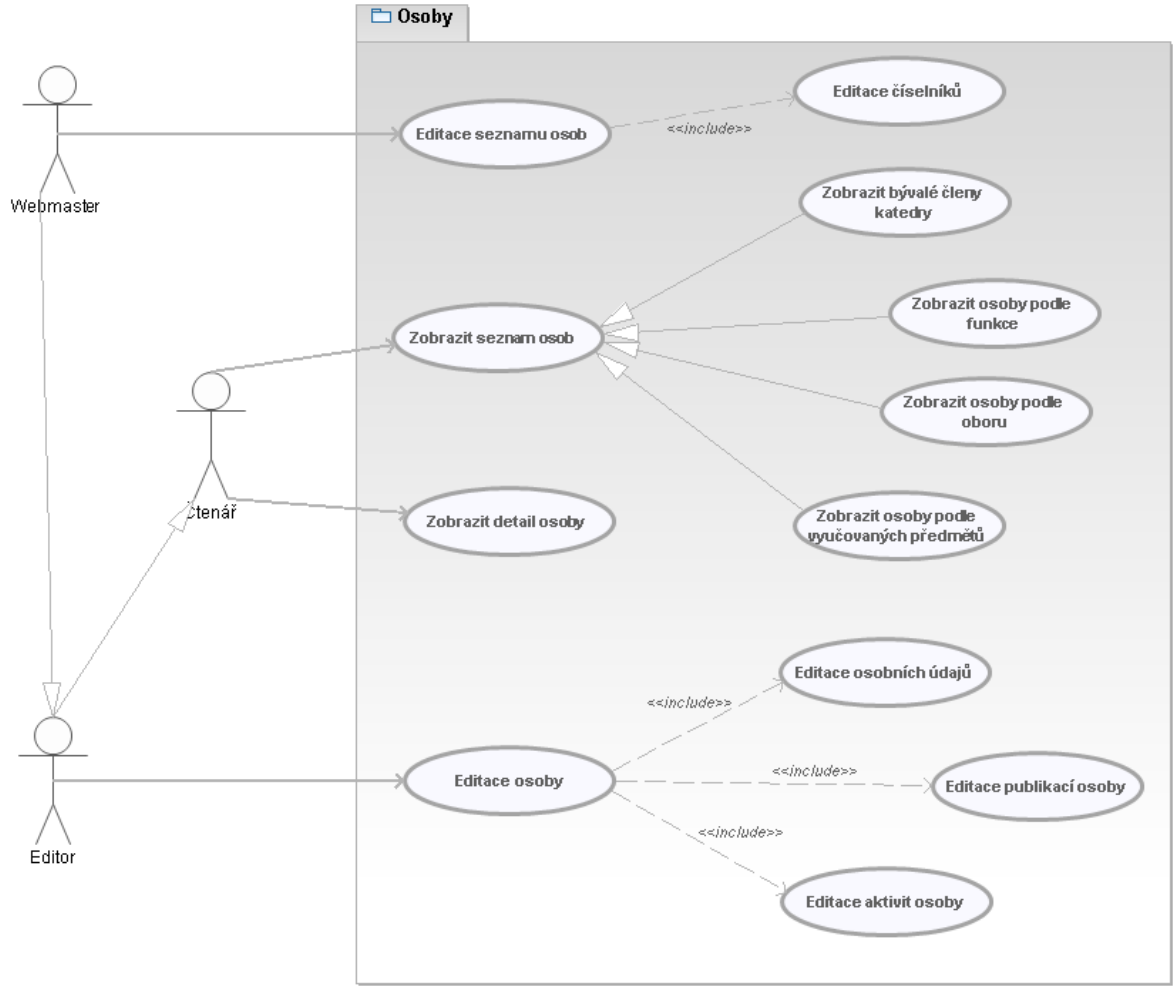
pozicemi, aktivitami, členství osob. Dále umožnit zařazení pozic do kategorií, přiřazení pozic, aktivit a členství osobám.

6.3.2 Role uživatelů modulu a případy užití

V případech užití týkajících se tohoto modulu rozlišuji následující role uživatelů:

- **Webmaster** – Na stránkách Osob má veškerá práva. Může prohlížet, provádět veškeré CRUD operace nad osobami a jejich číselníky.
- **Editor –** Na rozdíl od Webmaster role může editovat v osobách pouze svůj detail včetně přiřazování číselníků. Může cokoliv prohlížet.
- **Čtenář –** Jakýkoliv jiný uživatel, který zavítá na stránky. Může je pouze prohlížet.

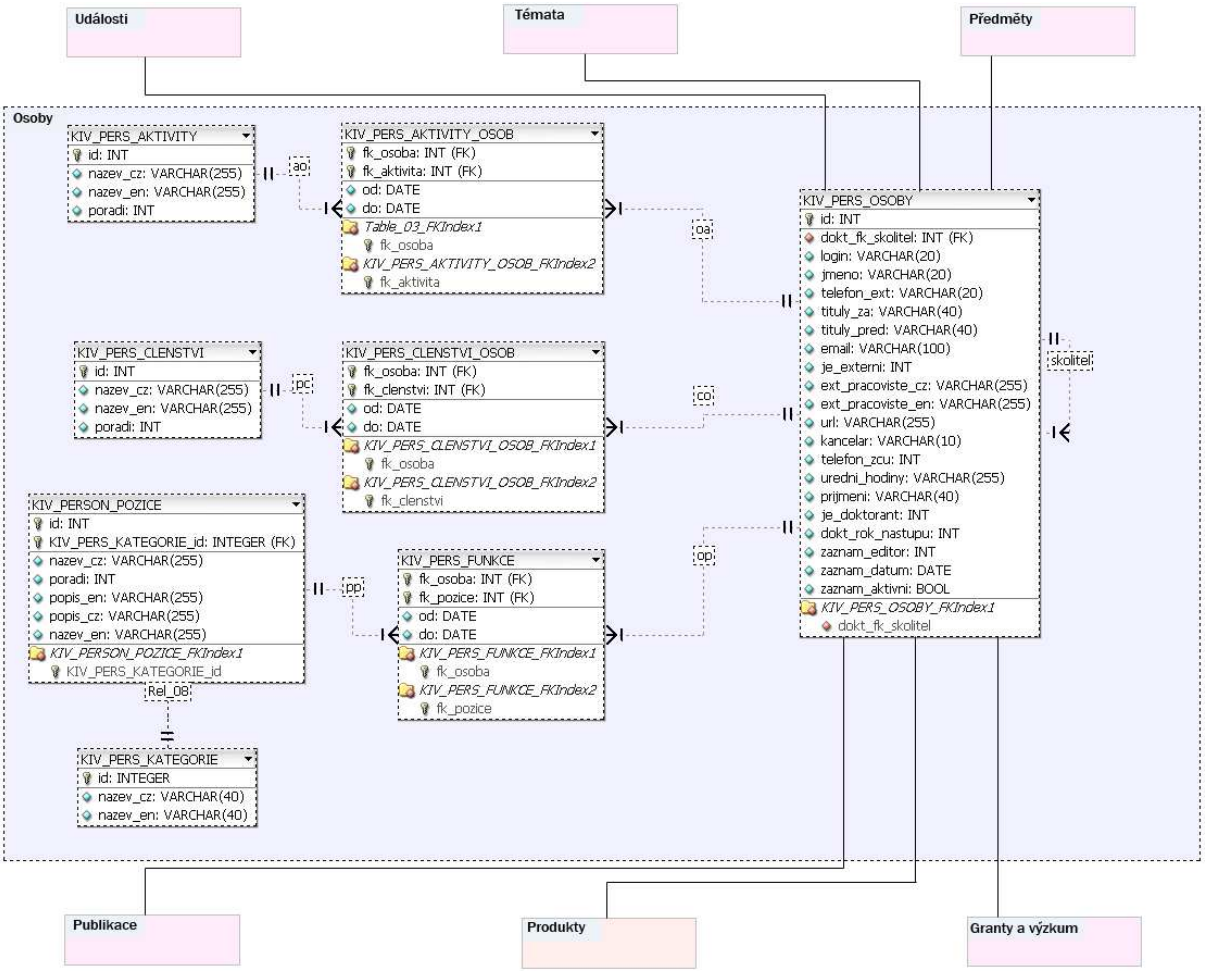
Případy užití k tomuto modulu ilustruje následující Obr-6.1.



Obr-6.1 – Případy užití modulu Osoby.

6.3.3 Datový model modulu

Datový model modulu osoby znázorňuje ERA diagram na Obr-6.2. Jedná se o část datového modelu pro celý web katedry, který je dostupný na [19], ve které jsou alespoň naznačeny návaznosti na části datového modelu pro další moduly. Návrh datového modelu nebyl součástí této práce.

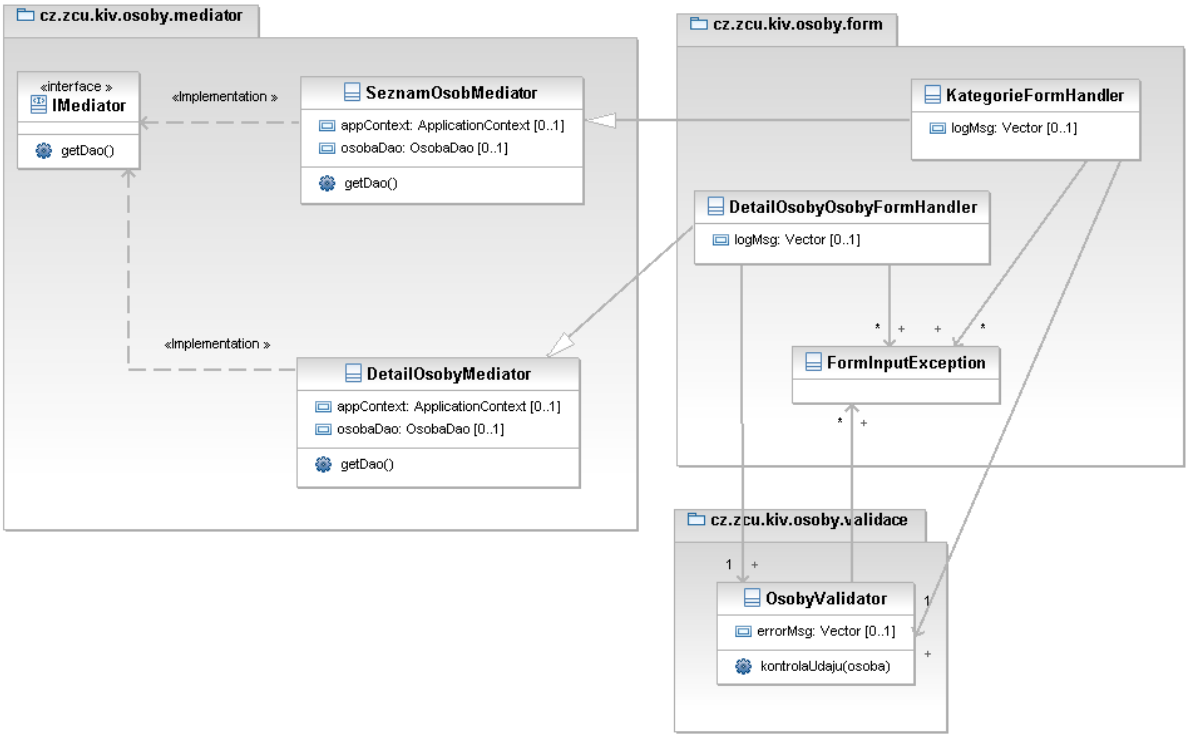


Obr-6.2 – Datový model pro modul Osoby.

6.3.4 Aplikační vrstva modulu

Implementace aplikační vrstvy modulu je zahrnuta v JAR knihovně *cz.zcu.kiv.osoby.jar*, která je součástí modulu a nachází se v jeho adresáři *lib*.

V této kapitole popíši funkčnost balíčků, které jsou součástí knihovny. Ty jsou znázorněny i se svými třídami v diagramu tříd na Obr-6.3.



Obr-6.3 – Diagram tříd JAR knihovny z modulu Osoby

6.3.4.1 Package cz.zcu.kiv.osoby.mediator

Balíček obsahuje všechny třídy typu Mediator v modulu, které zprostředkovávají data z DAO objektů souvisejících s datovou vrstvou pro modul osob. K dispozici jsou dvě Mediator třídy, jejichž základ je zobrazen na Obr-6.3. Mimo toho obsahují ještě několik metod pro zprostředkování operací typu „Read“ z DAO vrstvy.

- *SeznamOsobMediator* – Soustředí se na získávání dat potřebných pro vytvoření seznamu osob i s přihlédnutím na různé filtrování na základě pozic a kategorií osob. Po DAO objektech tedy žádá seznam pozic, kategorií, všech osob, dále se dotazuje na kategorie, pozice dle identifikátoru a na pozice dle kategorie, do kterých jsou zařazené atd. Získaná data ještě dále připravuje pro zobrazení na prezentační vrstvu.
- *DetailOsobyMediator* – Z DAO získává data potřebná pro vytvoření stránek s detailními informacemi o osobě, které zahrnují i data z číselníků aktivit a členství osob. Obsahuje tedy metody dotazující se na osobu dle identifikačního čísla, na seznam jeho aktivit a členství a dále na aktivity a členství dle jejich identifikačních čísel. Také využívá některé metody z Mediator třídy *SeznamOsobMediator*. Získaná data opět ještě upravuje před posíláním na prezentační vrstvu.

6.3.4.2 Package cz.zcu.kiv.osoby.form

Balíček obsahuje všechny třídy typu Form handler k tomuto modulu, které zprostředkovávají operace potřebné pro modifikaci datové vrstvy z DAO objektů, které souvisí s modulem osob.

- *KategorieFormHandler* – Zahrnuje operace pro modifikaci tabulek *KIV_PERS_KATEGORIE* a KIV_PERS_POZICE, které jsou potřebné ve spojitosti s filtrováním seznamu osob. Zprostředkovává operace pro uložení a smazání kategorie, nebo pozice. Před samotným uložením zpracovává data z prezentační vrstvy, tak aby byla připravena k persistenci.
- *DetailOsobyFormHandler* – Zahrnuje operace pro modifikaci jednotlivých osob, aktivit a členství. Nabízí se samozřejmě operace pro uložení a smazání osob, aktivit a členství. Dále pak přiřazení a odebrání aktivit a členství osobám. Opět před samotnou persistencí nejprve zpracovává data přicházející z prezentační vrstvy.

6.3.4.3 Package cz.zcu.kiv.osoby.validace

Balíček obsahuje jedinou třídu *OsobyValidator*, která poskytuje třídám typu Form Handler metody pro validaci údajů zadaných uživatelem ve formulářích prezentační vrstvy. Pokud třída detekuje uživatelem nesprávně zadané položky, vyhodí výjimku *FormInputException*, která pak bude třídami typu Form handler propagována na prezentační vrstvu.

Třída validátoru obsahuje např. metody pro kontrolu povinných údajů u osob, členství, aktivit, kategorií i pozic.

6.3.5 Prezentační vrstva modulu

Prezentační vrstvu pro modul osob přestavují JSP stránky umístěné uvnitř modulu v adresáři *pages*. Jejich výčet a stručný popis zahrnuje Tab-6.1. Běžné elementy stránek, které jsou využité na několika místech prezentační vrstvy, lze najít v JSP stránkách v adresáři *elements.* Ukázky jednotlivých stránek jsou k nahlédnutí v Příloze A, na obrázkách Obr-A.6 – Obr-A.11.

JSP stránka	Popis
seznam.jsp	Stránka umožňuje zobrazit seznam osob podle různých kategorií. Kategorie pak obsahují pozice, podle kterých jsou osoby seřazeny.
detail.jsp	Stránka zobrazuje detail osoby, jejíž login name je uložen v request parametru. Součástí detailu jsou základní informace o osobě vzhledem ke katedře, seznam jejích aktivit a členství, popř. seznam přiřazených doktorandů ke školení, nebo přiřazeného školitele.
edit.jsp	Stránka umožňuje editaci základních údajů osoby, jejíž login name je uložen v request parametru (Pokud request parametr není nastaven, vytvoří se nová osoba). Součástí stránky je také označení osoby jako externí s možností vyplnění doplňujících údajů a označení osoby jako doktoranda a přiřazení školitele.
ciselniky-edit.jsp	Stránka umožňuje vytvářet a mazat záznamy v číselníkových tabulkách osob. To znamená aktivity, členství, pozice, kategorie (pozice je navíc nutné zařadit do určité kategorie). U jednotlivých položek také uvádí odkaz na stránku pro jejich editaci.
aktivity-edit.jsp	Stránka umožňuje vytvářet a mazat aktivity a navíc je možné přiřazovat/odebírat aktivity osobě, jejíž login name je uložen v request parametru.
clenstvi-edit.jsp	Stránka umožňuje vytvářet a mazat členství a navíc je možné přiřazovat/odebírat členství osobě, jejíž login name je uložen v request parametru.
kategorie-edit.jsp	Stránka umožňuje vytvářet a mazat kategorie a navíc poskytuje možnost zařadit do kategorie nové pozice, popř. odebírat přiřazené.
pozice-edit.jsp	Stránka umožňuje přiřazovat/odebírat pozice osobě, jejíž login name je uložen v request parametru.

Tab-6.1 – JSP stránky modulu Osoby

6.3.6 Možná rozšíření modulu

V této kapitole zmíním několik dalších funkčností, které by mohly být předmětem dalšího vývoje modulů. Některé z nich pak budou nezbytným doplňkem - Jedná se např. o funkčnosti, které jsou závislé na implementaci dalších modulů.

- Napojit modul Osoby na modul Publikace – Na detailu osoby pak bude možné navíc evidovat i publikační činnost zaměstnanců.

- Napojit modul Osoby na modul Předměty – Na detailu osob pak bude možné navíc evidovat i jimi vyučované předměty.
- Napojit modul Osoby na modul Produkty – Na detailu osob pak bude možné navíc evidovat i jimi vytvořený autorizovaný software.
- Evidovat u osob zahraniční pobyty.

6.4 Modul Produkty

Modul Produkty doplňuje web katedry o funkčnosti spojené s evidencí autorizovaného softwaru vytvořeného na katedře a s jeho poskytováním ke stažení návštěvníků webu. Autorizovaný software katedry bude v tomto popisu nazýván produktem.

6.4.1 Požadavky na modul

Hlavním cílem modulu je umožnit návštěvníkům webu katedry stáhnout zveřejněný autorizovaný software a to pouze po souhlasu s licencí, se kterou je software distribuován. Modul dále bude poskytovat následující funkčnosti spojené s hlavním cílem:

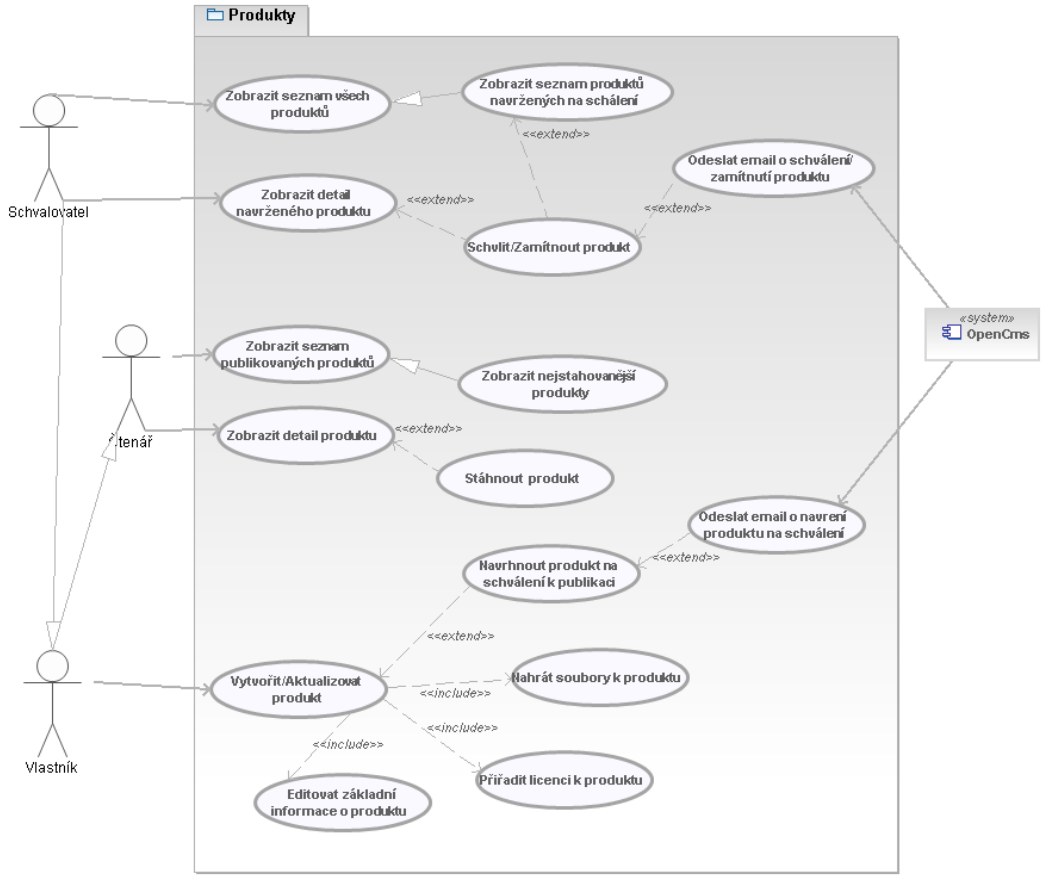
- Stránka reprezentující detail produktu musí zahrnovat soubory samotného softwaru, dokumentaci a licenci, se kterou je produkt distribuován.
- V modulu bude evidován počet stažení konkrétního produktu.
  - Na základě počtu stažení bude možné zobrazit seznam produktů seřazený podle počtu stažení.
- Na webových stránkách umožnit vytvoření, mazání (vyřazení) a editaci produktů (včetně přiřazení dostupných licencí).
- Vytvořené produkty budou před publikováním nejprve navrženy na schválení k publikaci
  - Implementovat systém schválení produktů – Zodpovědná osoba reviduje obsah produktu se vším, co k němu patří a rozhodne o tom, zda může být produkt zveřejněn. Pokud nebude produkt obsahovat vše, co má obsahovat, bude jeho návrh na publikaci odmítnut.
  - Aplikovat automatické generování a odesílaní emailů příslušným osobám při navržení produktu na schválení, schválení, odmítnutí.

6.4.2 Role uživatelů modulu a případy užití

Z hlediska přístupových práv na jednotlivé stránky se v modulu rozlišují tři role uživatelů.

- **Čtenář** – Jedná se o libovolného nepřihlášeného uživatele, který má právo prohlížet a stahovat publikované produkty.
- **Vlastník** – Jedná se o libovolného člena katedry, který je oprávněn vytvářet a editovat produkty.
- **Schvalovatel** – Jedná se o zodpovědnou osobu na katedře, která je oprávněna vytvářet, editovat, mazat a schvalovat produkty k publikaci.

Případy užití k modulu Produkty jsou znázorněné v diagramu na Obr-6.4.



Obr-6.4 – Případy užití modulu Produkty

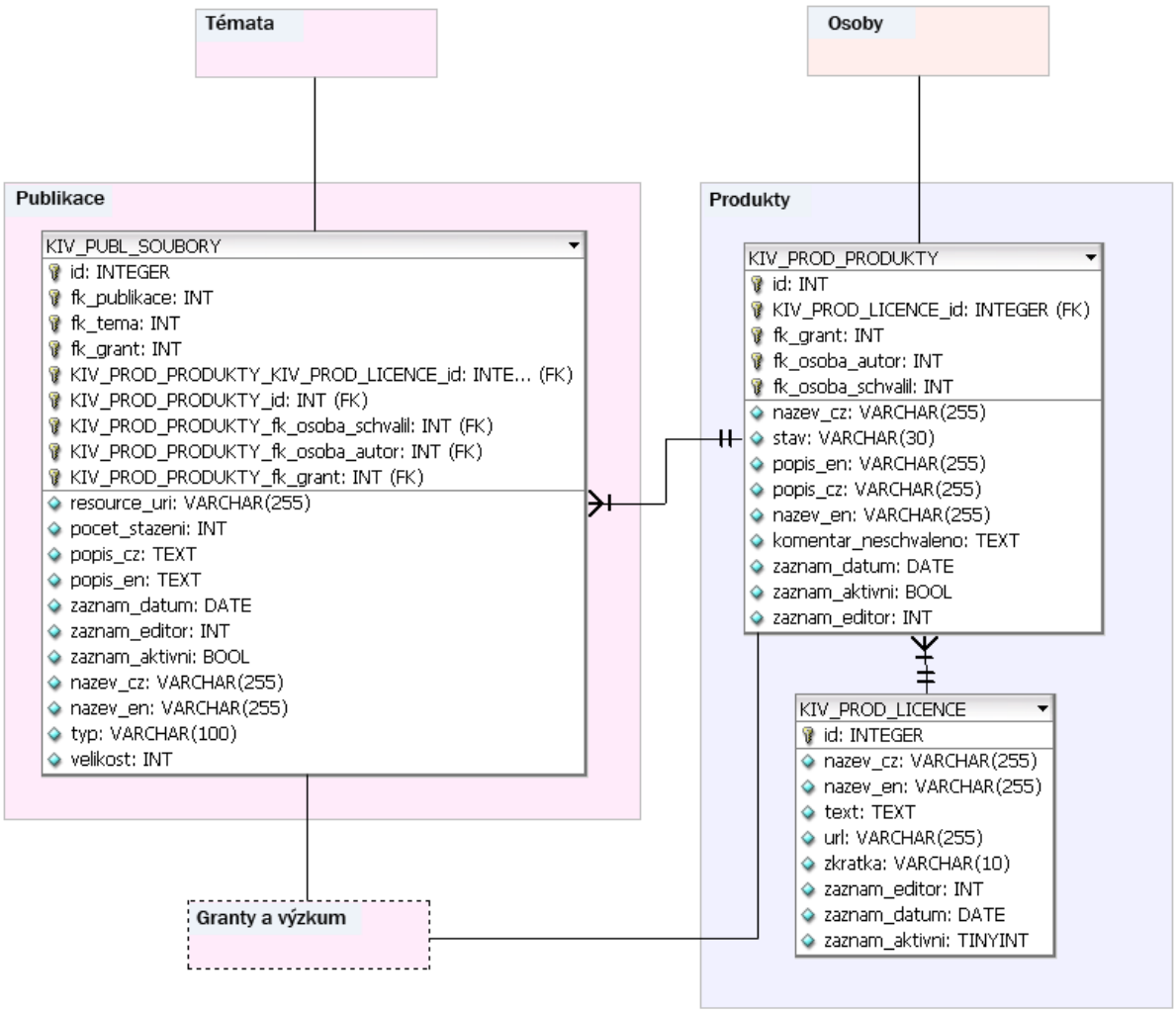
6.4.3 Stavy produktu

Každý produkt se může nacházet v jednom z následujících stavů:

- **Uložen** – Produkt je pouze vytvořen v systému, není publikován a nemusí zatím obsahovat žádné soubory. Musí však obsahovat povinné položky produktu – *Název CZ a Název EN*.
- **Navržen** – Produkt byl navržen na schválení. Čeká na schválení/zamítnutí schvalovatele. V tomto stavu již musí být připraven k publikaci, tzn., že musí obsahovat alespoň jeden soubor.
- **Schválen** – Produkt je již schválen schvalovatelem a je publikován. Pouze produkt v tomto stavu je viditelný pro *Guests* a pouze v tomto stavu je umožněno stáhnout produkt.
- **Neschválen** – Produkt byl shledán schvalovatelem jako nepřipraven k publikaci a zamítnut. Musí být autorem upraven a znovu navržen ke schválení.
- **Vyřazen** – Produkty se fyzicky přes webové rozhraní nemažou, místo toho se jim nastaví tento stav. Produkty se z tohoto stavu mohou dostat zpět “do provozu” přes stav uložen.

6.4.4 Datový model modulu

Datový model modulu produkty znázorňuje ERA diagram na Obr-6.5. Jedná se o část datového modelu pro celý web katedry, který je dostupný na [19], ve které jsou alespoň naznačeny návaznosti na části datového modelu pro další moduly. Návrh datového modelu nebyl součástí této práce.



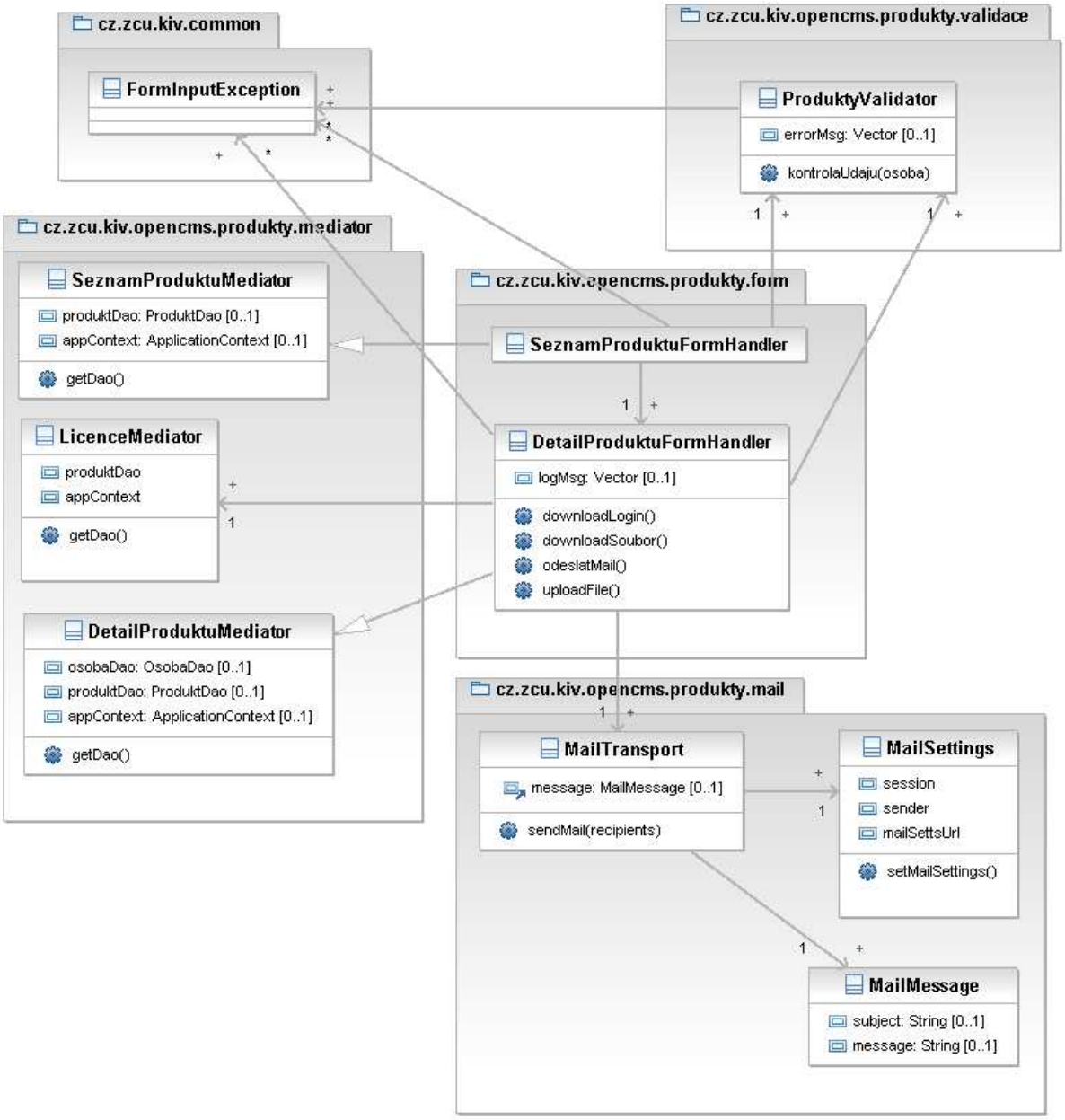
Obr-6.5 - Datový model pro modul Produkty

U tabulky *KIV_PUBL_SOUBORY* je třeba vysvětlit atribut *resource_url*. V tomto atributu je uložena URL resource objektu v OpenCms VFS, který představuje fyzický soubor pro záznam z této tabulky. Pokud tedy bude ke konkrétnímu produktu nahrán soubor, pak se nejprve nahraje do VFS a poté se navíc vytvoří záznam v této tabulce.

6.4.5 Aplikační vrstva modulu

Implementace aplikační vrstvy modulu je zahrnuta v JAR knihovně *cz.zcu.kiv.opencms.produkty.jar*, která je součástí modulu a nachází se v jeho adresáři *lib*.

V této kapitole popíši funkčnost balíčků, které jsou součástí knihovny. Ty jsou znázorněny i se svými třídami v diagramu tříd na Obr-6.6.



Obr-6.6 – Diagram tříd JAR knihovny modulu Produkty.

6.4.5.1 Package cz.zcu.kiv.opencms.produkty.mediator

Balíček obsahuje všechny třídy typu Mediator v modulu, které zprostředkovávají data z DAO objektů souvisejících s datovou vrstvou pro modul Produkty. K dispozici jsou tři Mediator třídy, jejichž základ je zobrazen na Obr-6.6. Mimo toho obsahují ještě několik metod pro zprostředkování operací typu „Read“ z DAO vrstvy.

- *SeznamProduktuMediator* – Z DAO objektů si nechává posílat seznamy produktů na základě jejich stavu, dle přihlášené osoby jako autora a dle počtu stažení. Produktů dle

přihlášené osoby se dožaduje, protože uživatel s rolí *Vlastník* může editovat pouze produkty, které sám vytvořil a díky této funkci bude mít vlastní produkty zobrazené na seznamu produktů v prezentační vrstvě.

- *DetailProduktuMediator* - Z DAO získává data potřebná pro vytvoření stránek s detailními informacemi o produktu. Mimo jiné se tedy dotazuje i na seznam souborů určitého produktu, na konkrétní soubory a produkty dle jejich identifikačního čísla. Získaná data ještě dále připravuje pro zobrazení na prezentační vrstvu. Tento mediator také kontroluje role přihlášeného uživatele a podle toho mu umožňuje přístup k produktům.
- *LicenceMediator* – Tento mediator je určen pro získávání licencí z DAO vrstvy.

6.4.5.2 Package cz.zcu.kiv.opencms.produkty.form

Balíček obsahuje dvě třídy typu Form handler, z nichž *SeznamProduktuFormHandler* poskytuje pouze dvě metody pro úpravu databáze, které jsou spojené se seznamem produktů. Jedná se o schválení a zamítnutí více produktů najednou. Mnohem důležitější třídou v kontextu celého modulu je *DetailProduktuFormHandler*.

DetailProduktuFormHandler již klasicky zprostředkovává stránkám spojeným s jedním konkrétním produktem operace pro modifikaci tabulek databáze týkajících se modulu Produkty. K dispozici jsou operace pro vytváření, editaci, mazání (vyřazení) jednotlivých produktů a souborů.

Dále tato třída zajišťuje následující důležité akce:

- **Nahrání resource objektu do VFS –** Data jsou z editačního formuláře posílána ve formě request parametru typu *multipart/form-data* kvůli možnosti nahrání souboru k produktu. Tento form handler pak tento typ request parametru zpracovává a příslušný soubor nahraje do OpenCms VFS.
- **Stažení souborů produktu –** Produkt může mít více souborů. Stažení produktu však znamená stažení jediného souboru. Form handler tedy při požadavku na stažení zabalí všechny soubory produktu do ZIP archivu a ten poskytne ke stažení.
- **Přihlášení uživatele, který má patřičná oprávnění k VFS složce se soubory produktů -** Nepřihlášený uživatel nemá přístup ke složce, v níž jsou uloženy soubory produktů. Důvodem je požadavek na stažení produktů pouze za předpokladu akceptování podmínek licence produktu. To znamená, že si uživatel nebude moci

stáhnout soubory ani již publikovaného produktu. Z tohoto důvodu bylo třeba vytvořit pomocného uživatele pro stáhnutí produktů. Systém pak při souhlasu s licencí přihlásí tohoto uživatele a po stáhnutí produktu jej opět odhlásí.

- **Odeslání emailu –** Využívá k tomu tříd z balíčku *cz.zcu.kiv.opencms.produkty. mail*. Odesílá emaily o navržení produktu všem schvalovatelům, nebo o schválení/zamítnutí produktu jeho autorovi.

6.4.5.3 Package cz.zcu.kiv.opencms.produkty.validace

Balíček obsahuje validační třídu pro modul Produkty. Její název je *ProduktyValidator* a poskytuje pro Produkty analogicky stejnou funkcionalitu jako validátor z modulu Osoby popsaný v kapitole 6.3.4.3.

6.4.5.4 Package cz.zcu.kiv.opencms.produkty.mail

Balíček zajišťuje odesílání emailových zpráv pro modul Produkty s využitím Java Mail API. Zahrnuty jsou třídy:

- *MailSettings* – Nastavuje parametry *mail session* pro odeslání emailu.
- *MailMessage* – Objekt emailové zprávy, která bude odeslána.
- *MailTransport* – Zajišťuje samotné odeslání emailu.

6.4.6 Prezentační vrstva modulu

Prezentační vrstvu pro modul Produkty přestavují JSP stránky umístěné uvnitř modulu v adresáři *pages*. Jejich výčet a stručný popis zahrnuje Tab-6.2. Běžné elementy stránek, které jsou využité na několika místech prezentační vrstvy, lze najít v JSP stránkách v adresáři *elements.* Ukázky jednotlivých stránek jsou k nahlédnutí v Příloze A, na obrázkách Obr-A.12 – Obr-A.16.

JSP stránka	Popis
seznam.jsp	Stránka zobrazuje seznam publikovaných produktů pro každého uživatele. Pro uživatele s rolí *Vlastník* navíc poskytuje seznam produktů, jejichž je vlastník autorem. Pro uživatele s rolí *Schvalovatel* jsou zobrazeny produkty se všemi stavy.
seznam-nejstahovanejsi.jsp	Stránka zobrazuje seznam publikovaných produktů seřazených sestupně podle počtu jejich stažení.

JSP stránka	Popis
detail.jsp	Stránka zobrazuje veškeré informace o produktu včetně přiřazené licence a souvisejících souborů. K dispozici je zde také zatržítko souhlasu s licencí a tlačítko pro stáhnutí produktu.
edit.jsp	Stránka umožňuje vytvoření/editaci produktu. Součástí stránky je formulářový prvek pro nahrání souboru k produktu a přiřazení licence. Uložený produkt lze odtud také odeslat ke schválení.
schvaleni.jsp	Stránka umožňuje schválit/zamítnout vybrané navržené produkty. Při zamítnutí produktu je možné přidat komentář s důvodem zamítnutí. Pro schvalovatele je zde i tlačítko pro schválení produktu.
produkt-download.jsp	Jedná se pouze o pomocnou stránku pro samotné stažení produktu. Její obsah je typu *application/zip* a obsahuje pouze soubory produktu ve formě byte pole ve své odpovědi.

Tab-6.2 – JSP stránky modulu Produkty

6.4.7 Možná rozšíření modulu

Tato kapitola opět zahrnuje výčet některých funkčností, které by mohli být předmětem dalšího vývoje modulu Produkty.

- Logovat stahování produktů. Log by měl obsahovat alespoň čas stažení, IP adresu uživatele, který stahoval a předmět stažení.
- Umožnit zamčení publikovaného produktu, tak aby nemohl být změněn ani jeho autorem. Odemknout produkt k editaci bude moci pouze Admin.
- Umožnit evidenci více verzí jednoho produktu.
- Umožnit autorovi produktu stáhnout produkt i bez souhlasu s licencí popř. umožnit mu stáhnout produkt v jakémkoli stavu (pokud obsahuje nějaké soubory).

6.5 Nasazení modulů této kapitoly

Moduly byly vyvíjeny a testovány za použití následujícího software a příslušných verzí:

- Windows Vista Business SP1 CZ 32bit,
- Java JDK 1.6.0_12,
- Apache Tomcat Version 6.0.14,

- MySQL 5.0.45-community-nt,
- OpenCms 6.2.3,
- Eclipse Platform Version 3.4.1,
- OpenCms Module Developer Eclipse plugin 0.0.5.

Oba moduly byly testovány průběžně během implementace na instalaci OpenCms, na které byly vyvíjeny. Zároveň finální verze modulů v rámci této práce byly nasazeny na testovací server na katedře KIV *beta.kiv.zcu.cz*. Testovací verzi modulu Produkty lze najít na adrese:

- http://beta.kiv.zcu.cz/produkty/

Na modulu Osoby již v současné době probíhají rozšiřující práce a v současné době není k dispozici jeho testovací verze online.

Modul Produkty byl navíc nasazen do ostrého provozu na *www.kiv.zcu.cz*. Jeho aktuální verze se nachází na adrese:

- http://www.kiv.zcu.cz/vyzkum/software/

7 Závěr

Jak bylo řečeno v úvodu této práce, přestavba katedrálního webu KIV na systém OpenCms probíhá již několik let. Já jsem se zapojil do tohoto projektu v rámci své bakalářské práce [8] ve fázi, kdy byl vybrán právě OpenCms jako systém pro správu obsahu, na kterém bude web postaven. Od té doby jsem byl součástí několika fází vývoje přestavby. Zúčastnil jsem se prvních pokusů o nasazení, vytváření prvních návrhů modulů pro rozšíření funkčnosti stránek a také menších úprav jádra systému, které mám již za ty tři roky práce se systémem z velké části zmapované. Z tohoto důvodu se hlavním cílem této práce stalo právě navržení a implementace rozšíření jádra OpenCms.

Jelikož je OpenCms open-source projekt, kladl jsem v práci důraz na zmapování současného i budoucího vývoje systému jak ze strany výrobců, tak ze strany katedry KIV a na základě toho byla v rámci plnění hlavního cíle vybrána rozšíření k implementaci. První plán zahrnoval implementaci tří rozšíření v rámci této práce. Jelikož se ale vybraná rozšíření ukázala složitější, než se původně očekávalo, přistoupilo se po domluvě k implementaci pouze dvou z nich. Požadavky na tato dvě rozšíření byly splněny a úspěšně otestovány.

Druhá část implementace se týkala OpenCms modulů pro doplnění funkčnosti webu KIV. Jednalo se o moduly „Osoby“ a „Produkty“. K jejich implementaci jsem přistoupil dříve než k plnění hlavního cíle, protože zvláště u modulu Produkty bylo hlavním požadavkem brzké nasazení do provozu. Z tohoto hlediska se podařilo splnit vytyčený cíl a oba dva moduly nasadit – Produkty do ostrého a Osoby do testovacího provozu. Jelikož se jedná o mnohem rozsáhlejší moduly než vybraná rozšíření jádra OpenCms, nebylo mým cílem pokrýt v rámci této práce veškeré požadavky na funkčnost. Navíc během půl roku testování modulů v provozu na ně vznikly další požadavky, týkající se opravy chyb a vylepšení funkčnosti. Další vývoj si tedy vzaly na starost jiné studentské týmy a v současné době se schyluje k vydání druhé verze obou modulů do provozu.

V teoretické části jsem se snažil o detailní popis implementačních detailů OpenCms, do kterého jsem zahrnul i zkušenosti, které jsem při práci s OpenCms nasbíral. Ty by měly

být přínosem pro budoucí i současné vývojáře, kteří se zapojí do projektu přestavby webu KIV a nejen pro ně.

Jelikož byla implementována pouze malá množina navržených vylepšení a ostatní byla doporučena k dalšímu vývoji, určitě se bude na katedře další vývoj ubírat i tímto směrem. Také aktuální seznam plánovaných modulů čítá několik položek, které se zajisté stanou předmětem dalších studentských prací na katedře. Nejprve bude ale určitě kladen důraz na aktualizaci systému OpenCms na poslední verzi, se kterou se zřejmě čeká na avizované vydání OpenCms 7.5 v červnu tohoto roku.

Přehled zkratek

ACE	Access Control Entry – Objekt třídy *CmsAccessControlEntry*.
ACL	Access Control List – Objekt třídy *CmsAccessControlList*.
API	Application programming interface.
CIV	Centrum informatizace a výpočetní techniky.
CMS	Content management system.
CRUD	Create, read, update, delete.
CSS	Cascading style sheets.
DAO	Data access object.
EL	Expression language.
HTML	Hypertext markup language.
IDE	Integrated development environment.
J2EE	Java 2 enterprise edition.
JAR	Java archiv.
JSF	Java server faces.
JSP	Java server pages.
JSTL	Java standard tag library.
KIV	Katedra informatiky a výpočetní techniky.
LGPL	Lesser general public licence.
RFS	Real file system.
TOC	Table of contents.
URI	Uniform resource identifier.
URL	Uniform resource locator.
VFS	Virtual file system.
WYSIWYG	What you see is what you get.
XML	Extensible markup language.
XSD	XML schema definition.

Literatura

[1] *CMS* [online]. 2007, [cit 2009-04-19]. URL <http://www.mediacentrik.cz/cz/slovnik/art_117/cms.aspx>.

[2] ROBERTSON, James. *Definition of information management terms* [online]. 2004, [cit. 2009-04-19]. URL <http://www.steptwo.com.au/papers/cmb_definition/index.html>.

[3] NEUMAJER, Ondřej. *Publikování na www: redakční a publikační systémy* [online]. 2004. 21s. URL <http://ondrej.neumajer.cz/download/cms.pdf>.

[4] KRUPIČKA, J. *Publikační systémy postavené na Java technologii*. Plzeň, 2006. 80 s. Diplomová práce na fakultě aplikovaných věd Západočeské univerzity na katedře informatiky a výpočetní techniky. Vedoucí diplomové práce Ing. Přemysl Brada, MSc., Ph.D.

[5] ROBERTSON, James. *What are the goals of a CMS?* [online]. 2002,[cit. 2009-04-19]. URL <http://www.steptwo.com.au/papers/kmc_goals/index.html>.

[6] BOIKO, Bob. *Content management bible*. 2nd edition, 2005. Wiley publishing, Inc. ISBN 0-7645-7371-3.

[7] ROBERTSON, James. *How to evaluate a content management system* [online]. 2002,[cit. 2009-04-19]. URL <http://www.steptwo.com.au/papers/kmc_what/index.html>.

[8] SKALICKÝ, S. *Doplňky redakčního systému OpenCms*. Plzeň, 2007. 67 s. Bakalářská práce na fakultě aplikovaných věd Západočeské univerzity na katedře informatiky a výpočetní techniky. Vedoucí diplomové práce Ing. Přemysl Brada, MSc., Ph.D.

[9] OpenCms development [online]. 2000. URL <http://www.opencms.org/en/development/>.

[10] OpenCms Wiki [online]. 2000. URL <http://opencms-wiki.org/>.

[11] OpenCms – Professional content management [online]. 2000. URL <http://www.opencms.org/>.

[12] Alkacon OpenCms 6.0 interactive documentation [online]. 2005. URL <http://beta.kiv.zcu.cz/doc>.

[13] jQuery plugin: Treeview [online]. URL <http://bassistance.de/jquery-plugins/jquery-plugin-treeview/>

[14] Alkacon Software – The OpenCms Experts [online]. 2003. URL <http://www.alkacon.com/>.

[15] The CMS Matrix [online]. URL <http://www.cmsmatrix.org/>.

[16] Apache Lucene [online]. 2006. <http://lucene.apache.org/java/docs/>.

[17] WEBBER, Christian. *Discussing OpenCms workplace usability* [online]. 2008, [cit. 2009-05-10]. URL <http://www.opencms.org/export/sites/opencms/en/events/opencms_days_2008/slides/B6_ChristianWeber.pdf>.

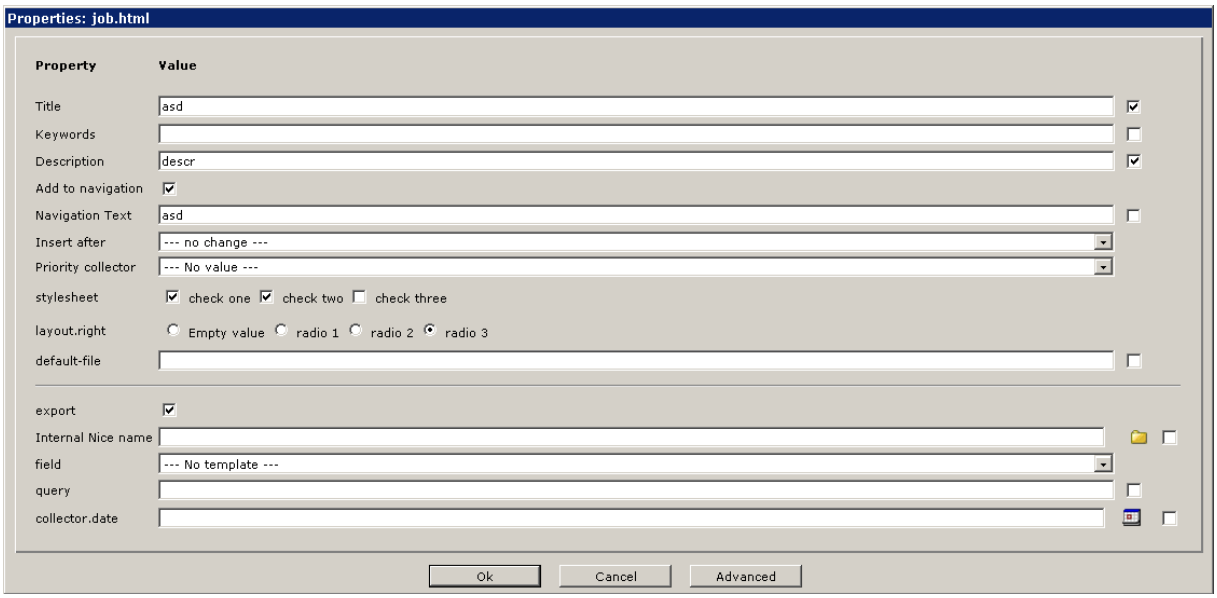
[18] OpenCms Forum [online]. 2006. URL <http://www.opencms-forum.de/opencms-forum/index>.

[19] Wiki KIVu – OpenCms [online]. URL <http://wiki.kiv.zcu.cz/OpenCMS/HomePage>.

Přílohy

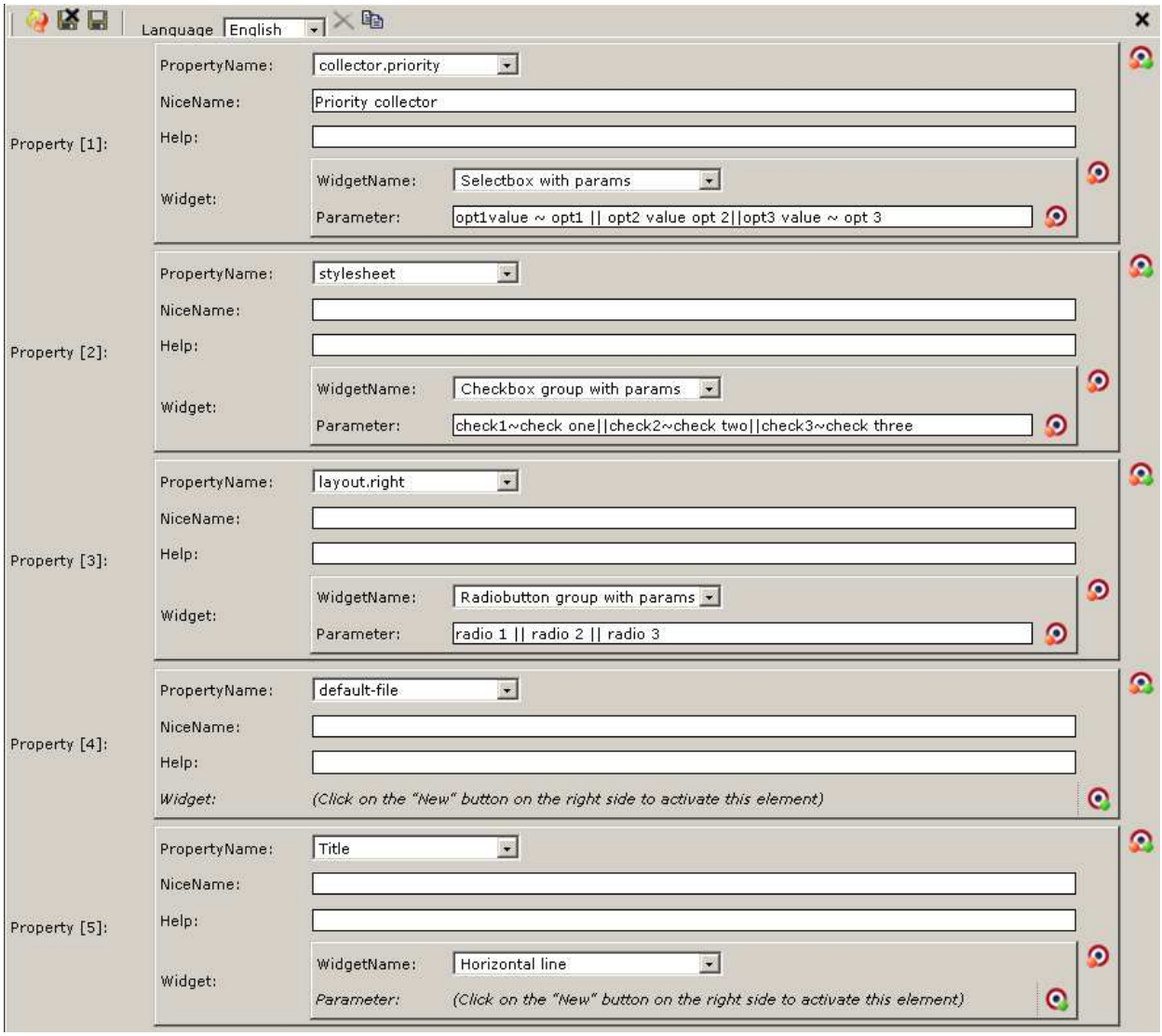
A. Ukázky implementovaných modulů

Modul „Parametrizovaný property dialog“



Obr-A.1 – Uživatelsky nastavený property dialog.

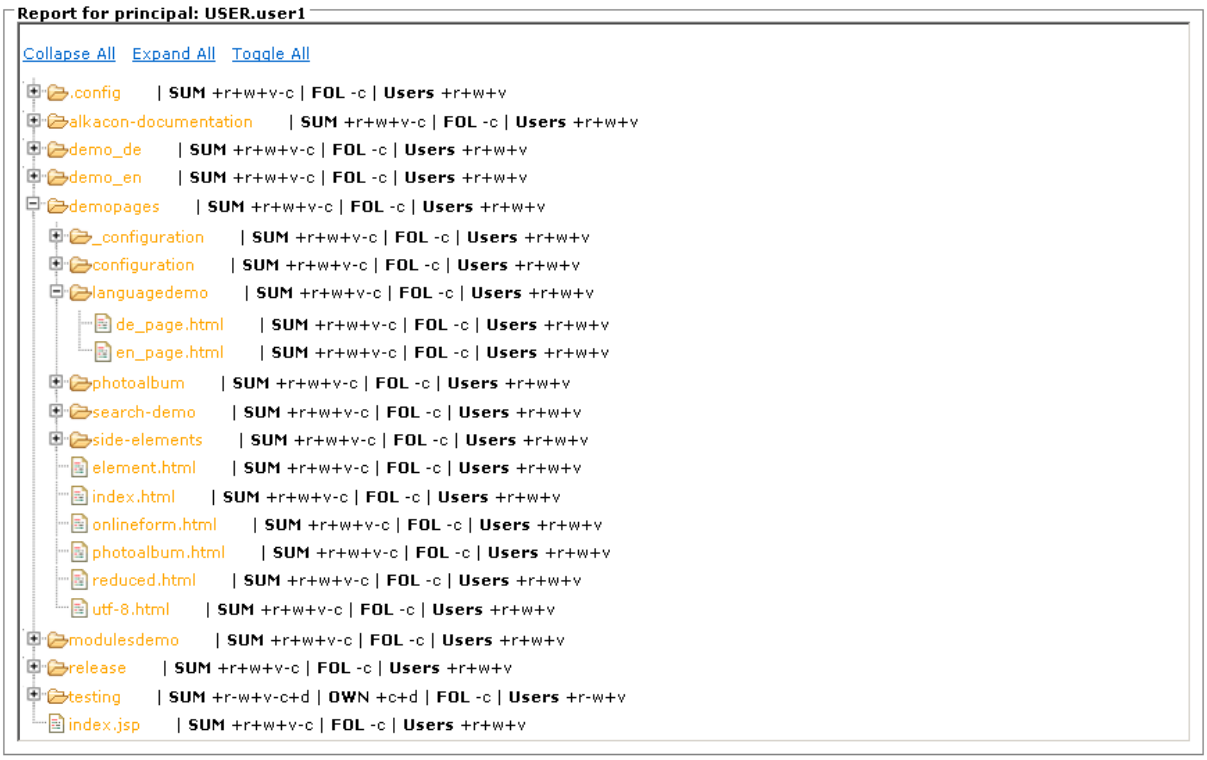
Obr-A.2 – Konfigurační soubor pro omezení výběru šablon.



Obr-A.3 – Konfigurace property dialogu.

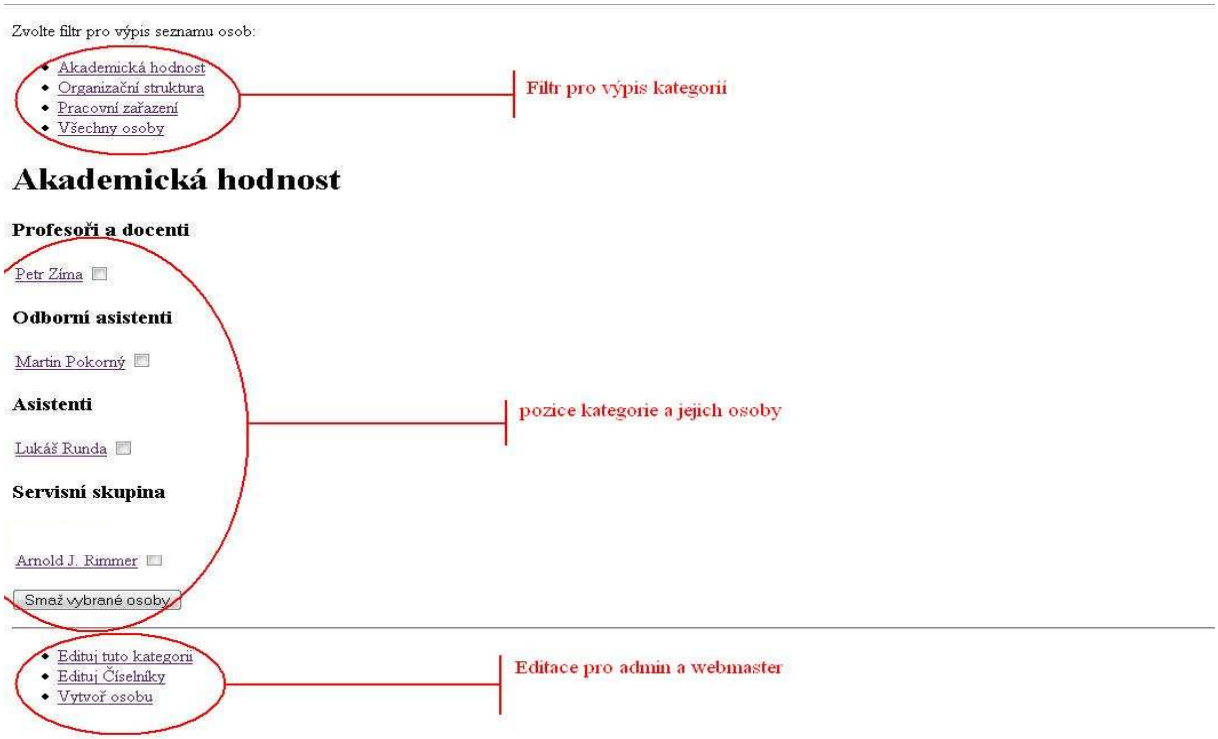
Modul „Report s přístupovými právy“

Obr-A.4 – Dialog s nastavením reportu.

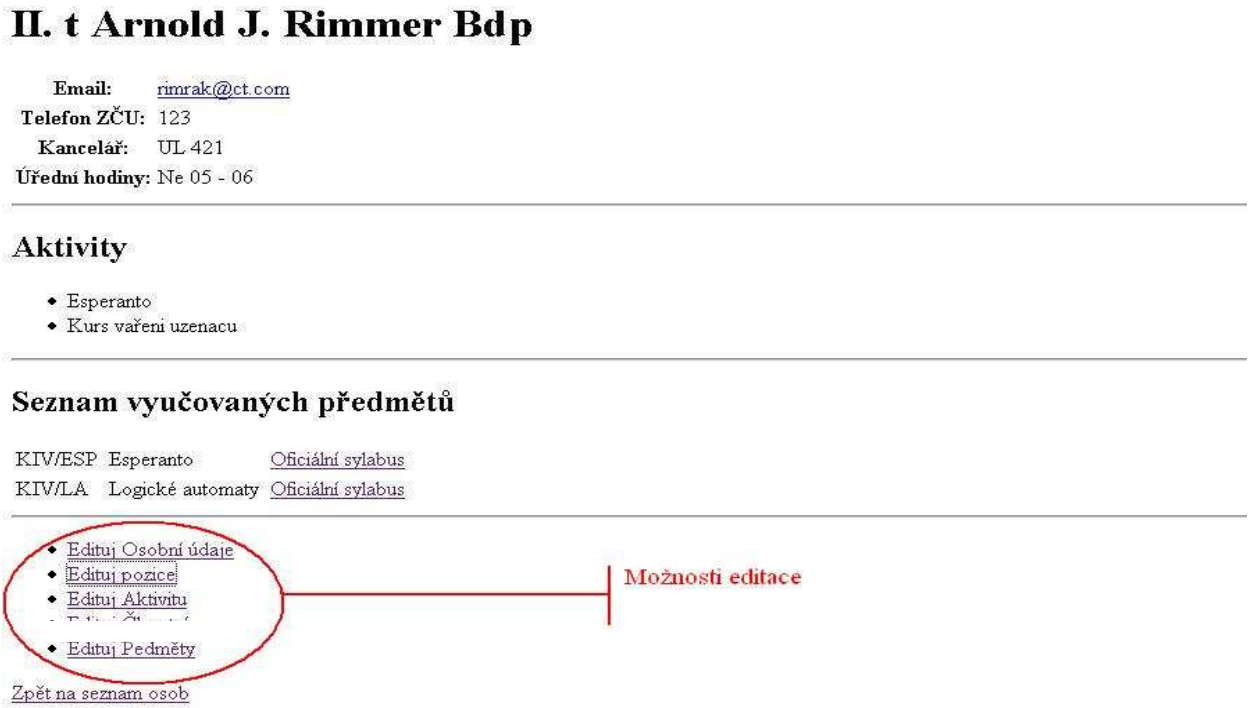


Obr-A.5 – Ukázka vygenerovaného reportu.

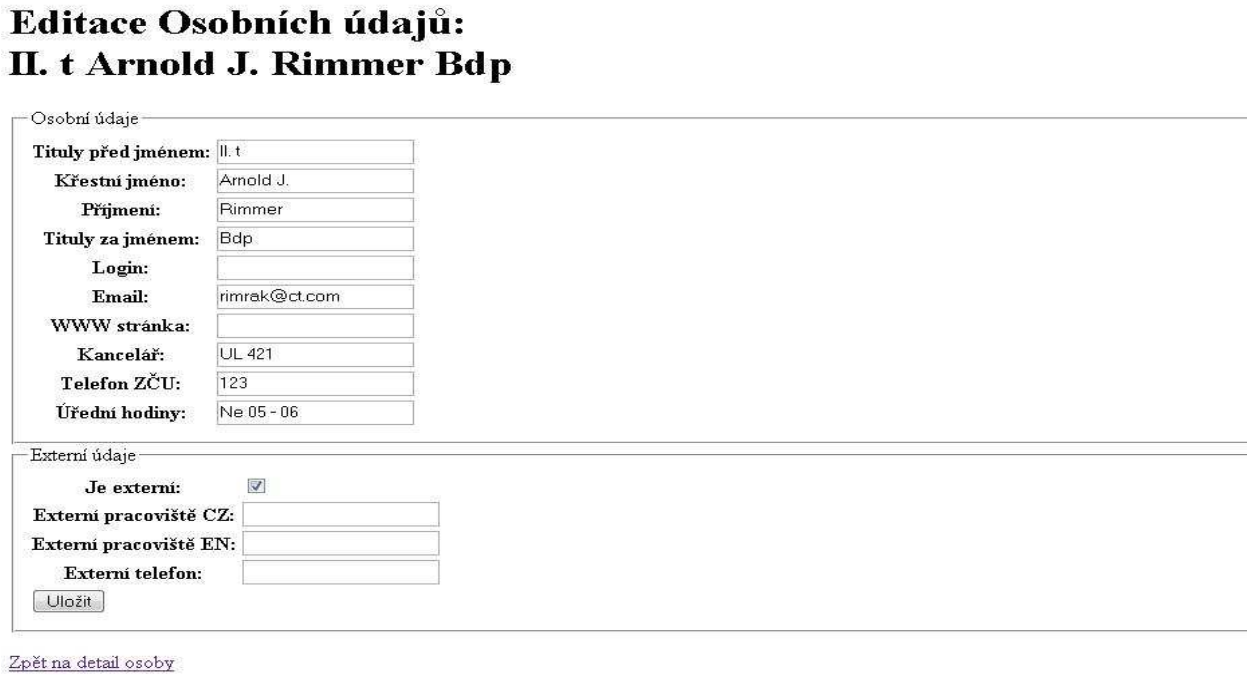
Modul Osoby



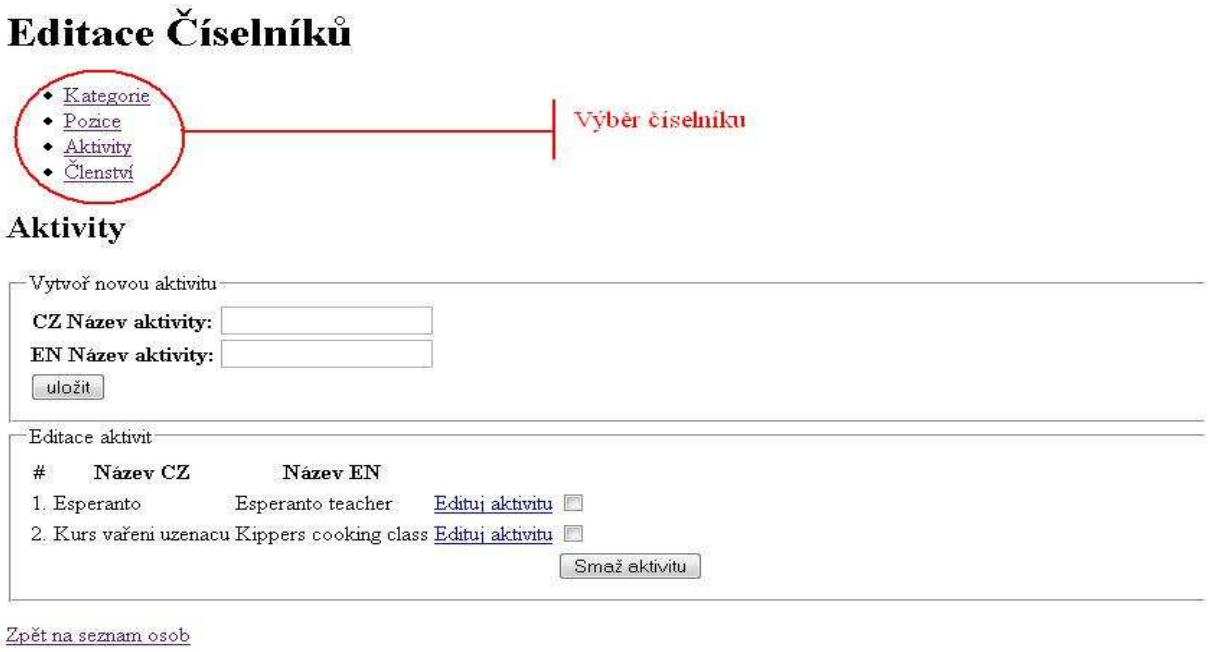
Obr-A.6 – Ukázka stránky se seznamem osob.



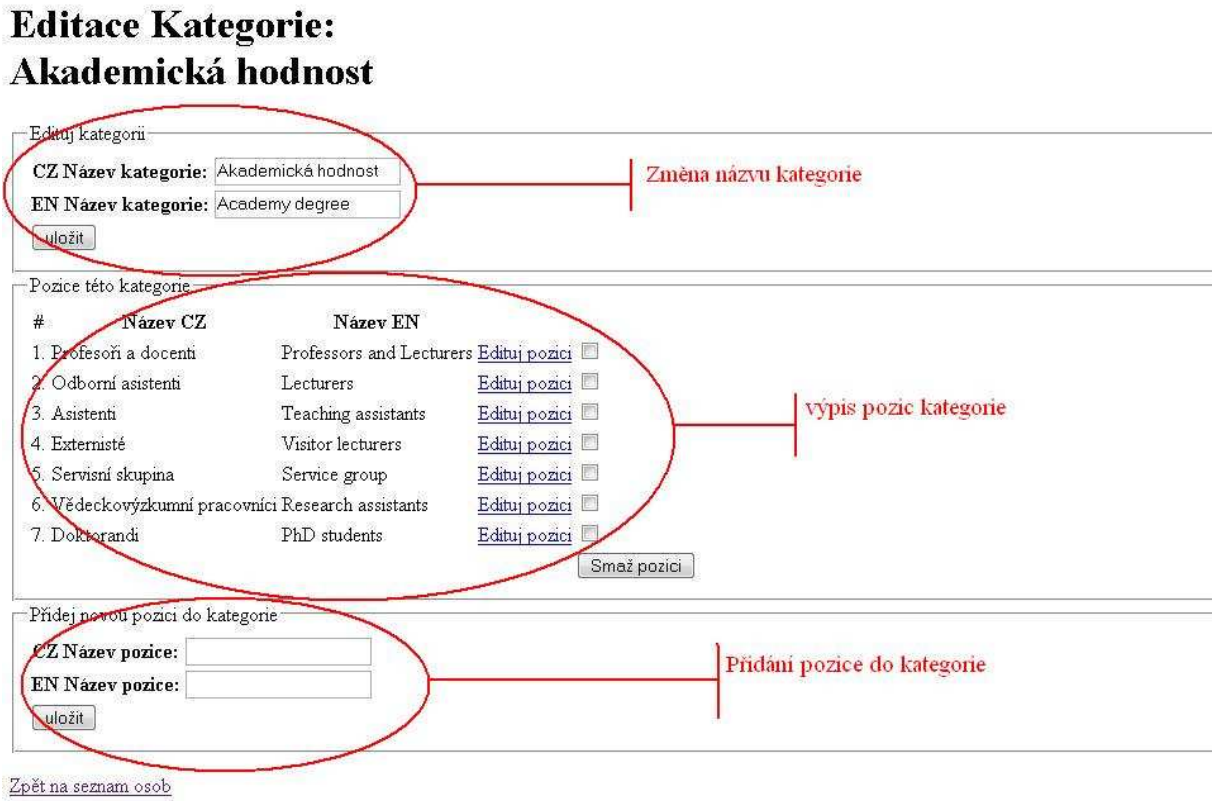
Obr-A.7 – Ukázka stránky s detailem osoby.



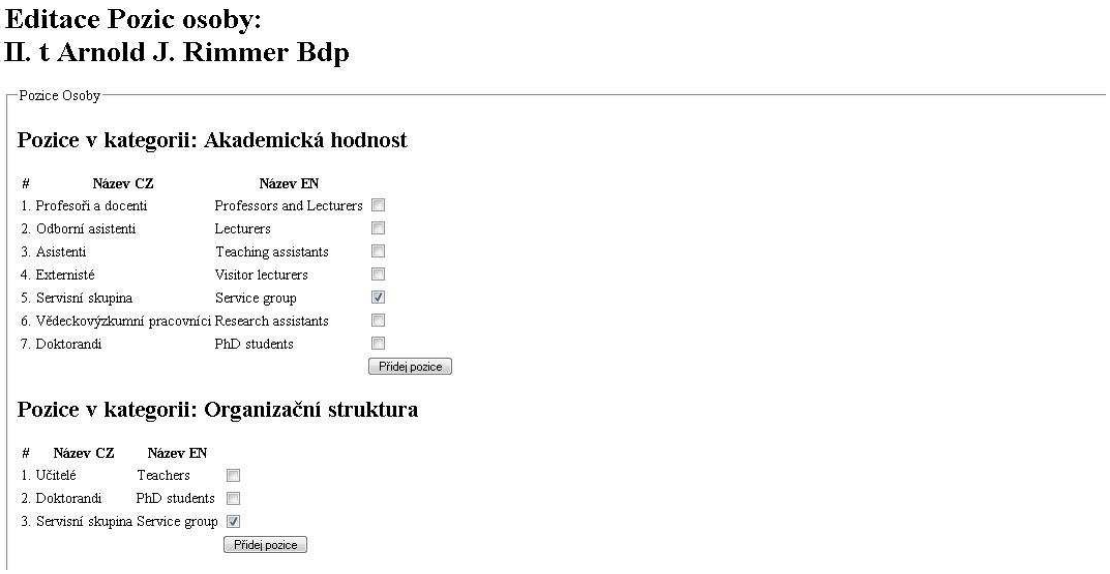
Obr-A.8 – Ukázka stránky s editací osobních údajů.



Obr-A.9 – Ukázka stránky editace číselníků.

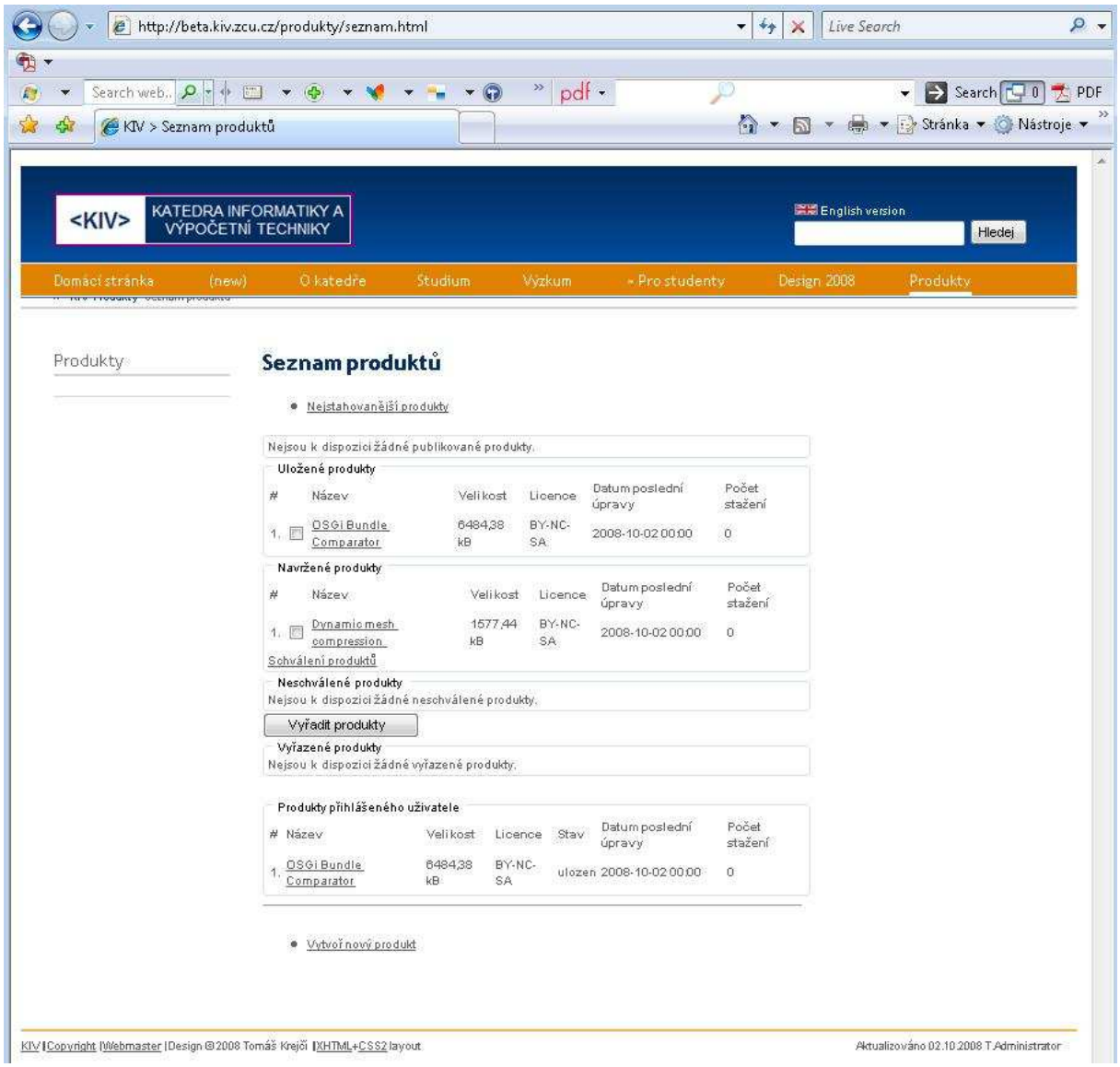


Obr-A.10 – Ukázka stránky editace kategorie.

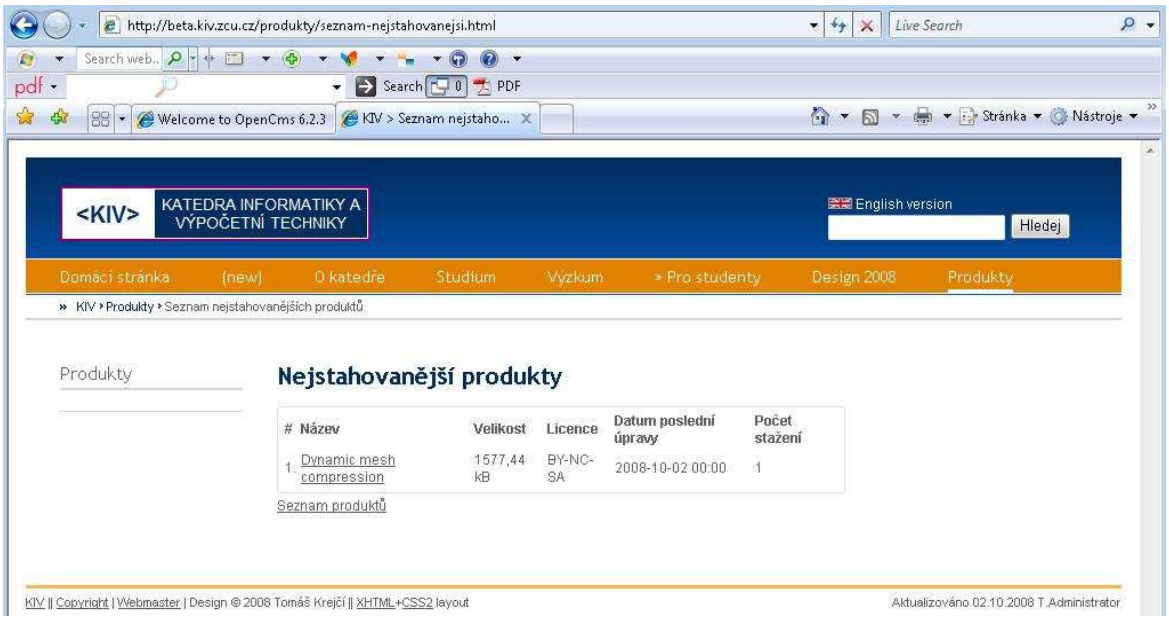


Obr-A.11 – Ukázka stránky editace pozic osoby.

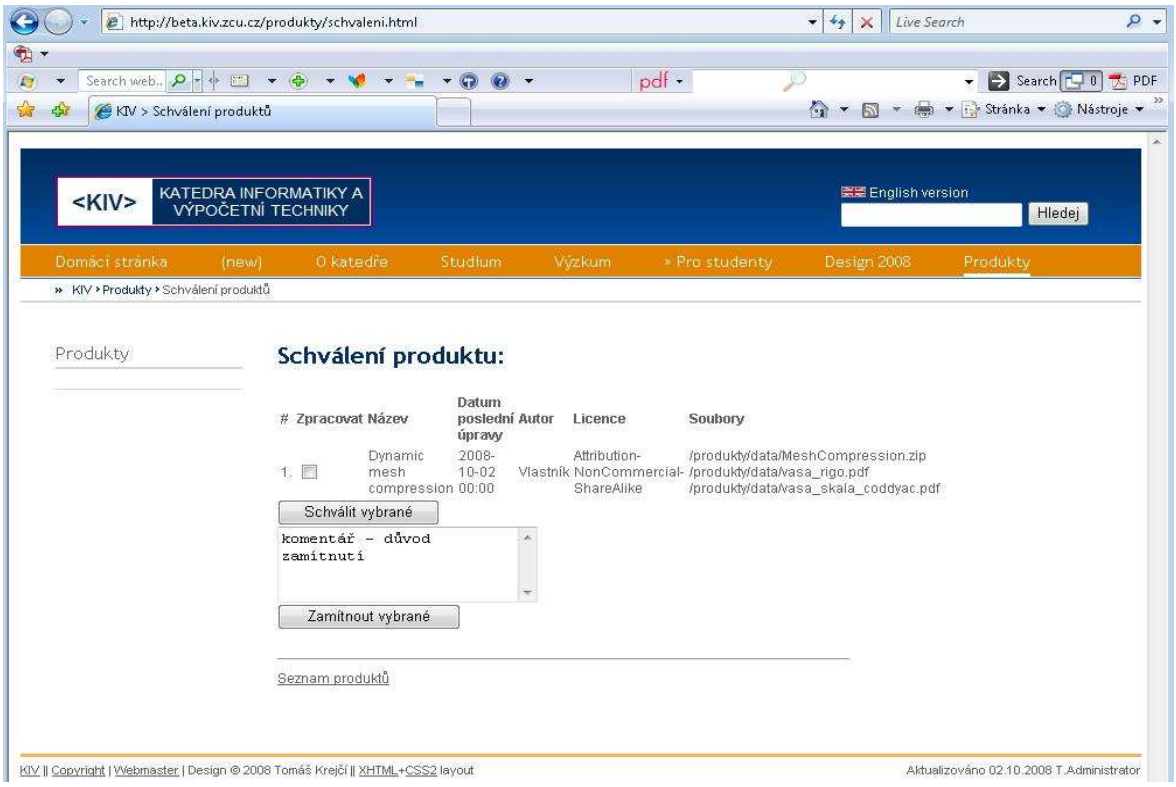
Modul Produkty



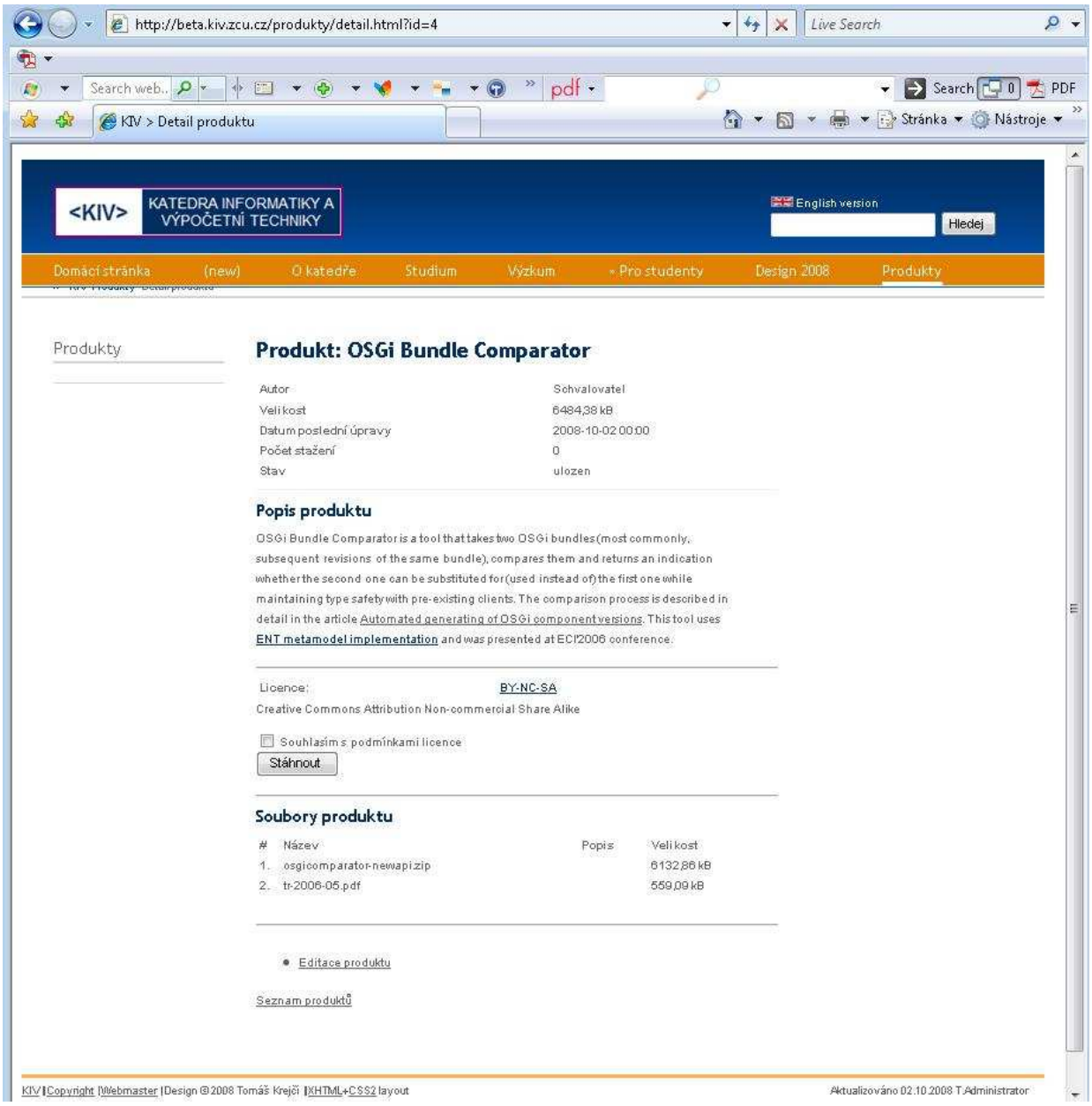
Obr-A.12 – Ukázka stránky se seznamem produktů.



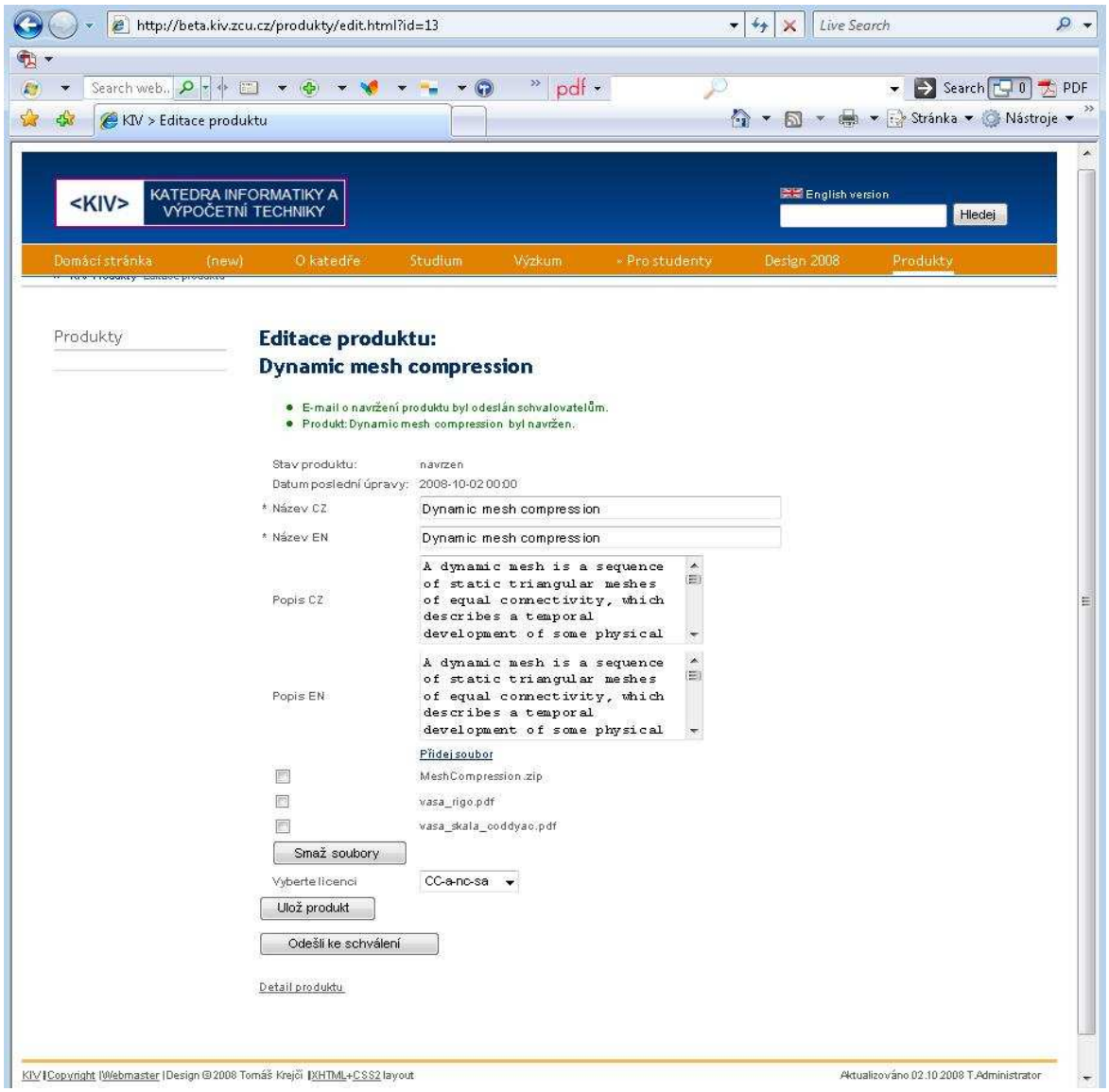
Obr-A.13 – Ukázka stránky se seznamem nejstahovanějších produktů.



Obr-A.14 – Ukázka stránky schválení produktu.



Obr-A.15 – Ukázka stránky s detailem produktu.



Obr-A.16 – Ukázka stránky s editací produktu.