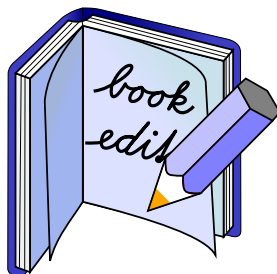


RichDoc Framework



Michal Šev enko <sevckenko@vc.cvut.cz> , 10. ledna 2007

Obsah

část I	Uživatelský návod	4
kapitola 1	Úvod	5
kapitola 2	Základy editace	9
kapitola 3	Editor rovnic	13
kapitola 4	Kreslení obrázků	20
kapitola 5	Managing Visual Styles	28
kapitola 6	Rejstřík, glosář a literatura	30
kapitola 7	Version Management and Localization	38
kapitola 8	Vytváření online kurzu	41
kapitola 9	Language Support	43
kapitola 10	Finding and Replacing Text	44
kapitola 11	Importování a exportování dokumentu	46
kapitola 12	Printing	55
kapitola 13	The ScratchPad Application	56
kapitola 14	Troubleshooting	57
kapitola 15	Acknowledgments	59
část II	Expert's Guide	60
kapitola 16	Introduction to the Data Model	61
kapitola 17	The RichDoc Print Format	62
kapitola 18	Contributing to the RichDoc Framework	68
kapitola 19	Styles	69
	Index	74
	Bibliography	77

Obsah

část I	Uživatelský návod	4
kapitola 1	Úvod	5

	1.1	Základní filozofie RichDoc Frameworku	5
	1.2	Co RichDoc Framework <i>není</i>	6
	1.3	Requirements and Installation	6
	1.4	Programy RichDoc Frameworku	6
	1.5	Začínáme	7
	1.6	Správa struktury dokumentu	7
kapitola 2		Základy editace	9
	2.1	Sekce dokument	9
	2.2	Přesouvání a kopírování objekt	9
	2.3	Formátování odstavce	10
	2.4	Klíčové odkazy	11
	2.5	Setting Properties of Objects	12
	2.6	Seznamy	12
	2.7	Tabulky	12
kapitola 3		Editor rovnic	13
	3.1	Matematický text	13
	3.2	Operátory a symboly	13
	3.3	Struktury	14
	3.4	Závorky	16
	3.5	Pole	17
	3.6	Mezery	17
	3.7	Zobrazené rovnice	17
	3.8	Úprava rovnic	18
kapitola 4		Kreslení obrázk	20
	4.1	Nový obrázek	20
	4.2	Přidáváme grafické objekty	20
	4.3	Editace obrázk	23
	4.4	Setting Visual Properties	24
	4.5	Mažení textu	24
	4.6	Transformace	25
	4.7	Animace	26
kapitola 5		Managing Visual Styles	28
kapitola 6		Rejstřík, glosář a literatura	30
	6.1	Úprava rejstříku a glosáře	30
	6.2	Úprava seznamu bibliografických citací	32
kapitola 7		Version Management and Localization	38
	7.1	Version Management	38
	7.2	Managing of Localized Documents	38
kapitola 8		Vytváření online kurz	41
	8.1	O standardu SCORM	41
	8.2	Creating SCORM Course Document	42
kapitola 9		Language Support	43
	9.1	Setting the Language	43
kapitola 10		Finding and Replacing Text	44
	10.1	Finding Text	44
	10.2	Replacing Text	45
kapitola 11		Importování a exportování dokument	46

	11.1	Rozhraní k export/import modul m	46
	11.2	Exporting HTML	47
	11.3	Exporting LaTeX	48
	11.4	Exporting PDF	50
	11.5	Exporting SCORM	51
	11.6	Importing HTML	51
	11.7	Importing LaTeX	53
	11.8	Importing DocBook	53
	11.9	Deploying Documents	54
kapitola 12		Printing	55
kapitola 13		The ScratchPad Application	56
	13.1	Managing the System of Notes	56
kapitola 14		Troubleshooting	57
	14.1	Platform-specific problems	57
	14.2	File Backup and Recovery	57
kapitola 15		Acknowledgments	59
část II		Expert's Guide	60
kapitola 16		Introduction to the Data Model	61
kapitola 17		The RichDoc Print Format	62
	17.1	The Overall Structure of a Print File	62
	17.2	The Page Description Format	62
	17.3	Notes on Printing to the RichDoc Print Format	66
kapitola 18		Contributing to the RichDoc Framework	68
	18.1	Contributing Localized Resources	68
kapitola 19		Styles	69
	19.1	Visual Styles	69
	19.2	Document Styles	71
	19.3	Bibliography Styles	71
		Index	74
		Bibliography	77

část I

Uživatelský návod

Kapitola 1

Úvod

RichDoc Framework je komplexní systém, který slouží k tvorbě, údržbě, výměně a prezentaci dokumentů, s dále na vdeckotechnické dokumenty. Systém je zatím experimentální, a vznikl v rámci projektu KSMSA. Původním úelem systému bylo ověřit výsledky autorovy disertační práce “Knowledge Support for Modeling and Simulation”, byl však zveřejněn pod GNU licenci, aby mohl být užitečný i ostatním případným uživatelům. V kapitole 14 jsou shrnuta všechna omezení systému. Rovněž jsou vítané jakékoli příspěvky do systému, viz kapitola 18.

1.1 Základní filozofie RichDoc Frameworku

Jaký je rozdíl mezi RichDoc frameworkem a podobnými systémy pro práci s dokumenty? Existují dva základní typy programů pro práci s dokumenty. Programy prvního typu, zvané též textové procesory, jsou založeny na principu WYSIWYG (What You See Is What You Get, co vidíš to dostaneš). Tento princip zajišťuje, že co uživatel vidí na obrazovce po práci s dokumentem je velmi podobné, ne-li stejné, finálnímu výsledku po vytisknutí dokumentu. Důsledkem tohoto principu je ale také to, že textové procesory jsou *prezentací-orientované* – soustředí pozornost uživatele na vzhled dokumentu, a poněkud zanedbávají péči o jeho strukturu. Na druhou stranu jsou však textové procesory velmi intuitivní, a proto s nimi mohou snadno pracovat i méně zkušení uživatelé.

Druhým typem programů jsou programy založené na tzv. značkovacích jazycích (markup languages), například LaTeX nebo HTML. Při práci s těmito programy obyčejně nevytváříme dokumenty vizuálně, ale pomocí textového editoru, kde zadáváme text dokumentu společně se speciálními značkami definujícími strukturální a vizuální aspekty dokumentu. Takové dokumenty jsou obvykle mnohem více *strukturně-orientované* – dokument primárně reprezentuje strukturu dokumentu, vizuální aspekty (typ a velikost písma, okraje atd.) je odvozen ze struktury dokumentu automaticky pomocí mechanismu, který je na dokumentu nezávislý. Takové oddělení umožňuje využití stejného dokumentu v různých kontextech, nebo struktura dokumentu je nezávislá na způsobu jeho prezentace. Jeden dokument tak může být například použit pro generování výstupu ve formátech HTML a PDF. Právně značkových dokumentů je však poněkud méně pohodlná, a vyžaduje znalost některého ze značkovacích jazyků, jež bývají poměrně složité.

My jsme se pokusili tyto přístupy zkombinovat, a vytvořili jsme systém, který je WYSIWYG jen například. Systém je WYSIWYG v tom smyslu, že při práci s dokumentem probíhá vizuálně, pomocí grafického uživatelského rozhraní. Systém však soustředí pozornost uživatele především na strukturu dokumentu, nikoli na jeho vzhled. Jemné vizuální aspekty finální prezentace dokumentu mohou být nastaveny dodatečně, bez zásahu do původního dokumentu. Struktura takto připraveného dokumentu je tedy mnohem více podobná struktuře dokumentu připravených pomocí značkovacích jazyků. Ve skutečnosti je za uživatelským rozhraním systému značkový jazyk, který je však navržen tak, aby výrazně podporoval vizuální způsob editace dokumentu.

RichDoc Framework tvoří kompaktní, přenositelnou softwarovou komponentu, která může být snadno integrována do složitějších softwarových systémů. Systém můžeme použít nejen pro tvorbu samostatných dokumentů jako jsou knihy a články, ale lze vytvářet i malé fragmenty dokumentace, které lze vkládat do různých nadřazených objektů, jako jsou záznamy v kalendáři, inženýrské soubory, a jakékoli objekty ke kterým je vhodné přidávat dokumentaci. To vše s podporou pokročilých funkcí jako je možnost vkládání seznamů, tabulek, matematických vzorců a grafiky.

1.2 Co RichDoc Framework není

RichDoc Framework má být užší využitelný i pro úpravu strukturovaných bohatých dokumentů, které však mají poměrně jednoduché mapování mezi strukturou dokumentu a jeho vizuální prezentací. Pokud například potěbujete často nastavovat vizuální vlastnosti částí dokumentu ad-hoc způsobem, může být použití RichDoc Frameworku pro tento účel příliš kostrbaté. Pokud vám uvedený scénář nevyhovuje, asi bude pro vás lepší použít některý z více vizuálně orientovaných textových procesorů, jako je například Microsoft Word. Systém také není vhodný pro úpravu dokumentů se složitým grafickým rozvržením částí dokumentu, například pro úpravu časopisů nebo novin.

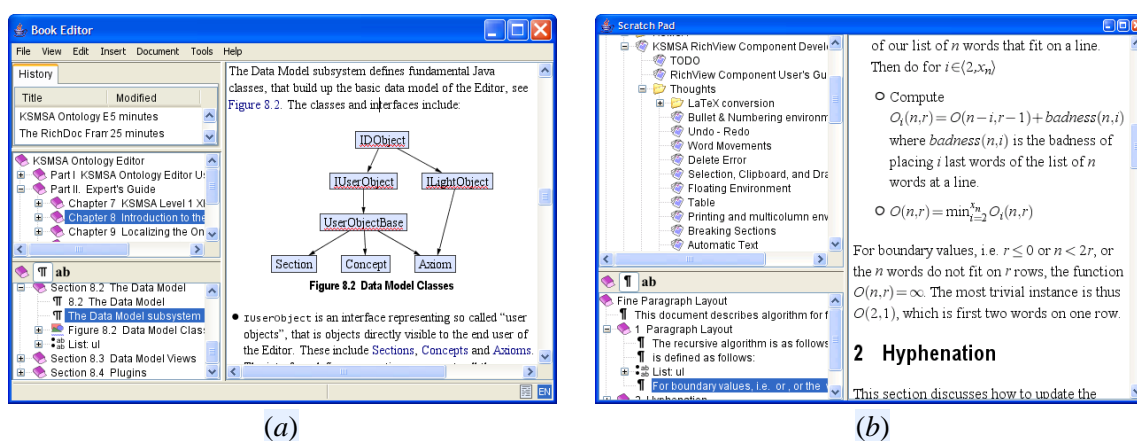
1.3 Requirements and Installation

The RichDoc Framework is pure Java application. It means that it runs on any platform that supports Java technology, including Microsoft Windows, many versions of the Unix operating system, and others. It has been tested on Microsoft Windows XP and RedHat Linux 9.1, but it is expected to run on other Java-enabled platforms as well.

Prior installation of the RichDoc Framework, you need to install the Java Runtime Environment (JRE) from Sun. Just visit <http://www.java.com> and download the JRE. If you have a JRE installed already, make sure that it is version 1.5 or later. If not, you should install the latest version.

Second step is to download and install a binary distribution of the RichDoc Framework. Two binary distributions are provided: Windows Installer to be installed on Windows, and compressed archive to be installed under UNIX. Installation under Windows is quite straightforward: just run the installer, and follow the instructions. After installation, the start menu is populated with appropriate shortcuts to run RichDoc applications. To install under UNIX, you need to unpack the distribution archive to an appropriate directory, and add the bin subdirectory to your PATH. Also make sure you have set the JAVA_HOME environment variable to the directory of your JRE installation. Then you can run any of the RichDoc programs by typing either bookEditor, scratchPad, or ioTool.

1.4 Programy RichDoc Frameworku

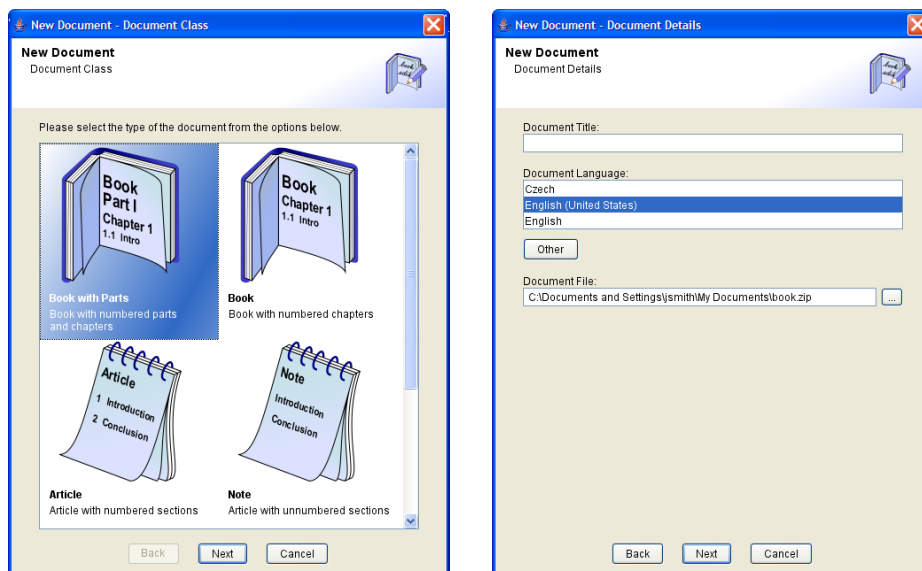


Obr. 1.1 Dva programy RichDoc Frameworku: (a) BookEditor, (b) ScratchPad

RichDoc Framework obsahuje dva základní programy. **BookEditor** je program, který slouží pro úpravu kompaktních, samostatných dokumentů, jako jsou knihy nebo vdecké články. Druhým programem je program **ScratchPad**, užitečný nástroj pro tvorbu osobních poznámek. Oba programy se liší ve způsobu organizace dokumentu a jejich obsahu, základní postupy pro úpravu dokumentů jsou však stejné. V tomto návodu budeme popisovat především program BookEditor. Program ScratchPad je popsán v samostatné kapitole 13.

1.5 Za ináme

P i práci s BookEditorem musíme za ít vytvo ením nového dokumentu. To provedeme p íkazem Soubor → Nový z hlavního menu, nebo stiskem Ctrl+N. Zobrazí se pr vodce vytvo ením nového dokumentu, viz obr. 1.2.



Obr. 1.2 Pr vodce vytvo ením nového dokumentu

V prvním kroku je třeba vybrat vhodnou *t ídu dokumentu*, která ur uje jeho celkový charakter. Dokument m že být typu **Kniha**, a **Kniha s ástmi**, **lánek** nebo **Poznámka**. Nastavení *t ídy* ovlivní mnoho vlastností dokumentu, jako je jeho vizuální styl, nebo zp sob íslování sekcí a obrázk . Lze vytvo vit i novou *t ídu*, což je však složitá operace nad rámec tohoto návodu.

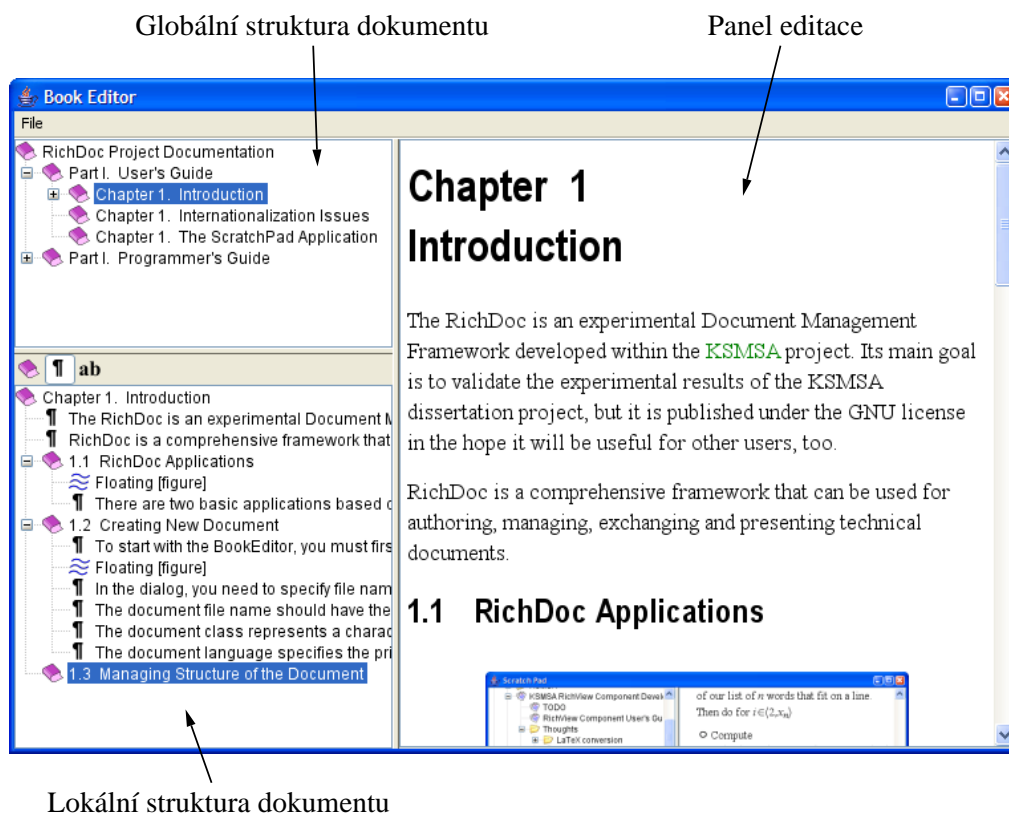
V druhém kroku zadáme název, jazyk a soubor dokumentu. *Jazyk dokumentu* ur uje hlavní jazyk použitý p ípsaní dokumentu. M žeme bu to vybrat jazyk z p ípraveného seznamu, nebo po stisku klávesy Jiný vybrat ze seznamu všech jazyk . P estože lze vybrat libovolný jazyk, v zte že pro mnoho jazyk m že program poskytnout malou nebo v bec žádnou podporu. V takovém p ípad se aplikují pravidla pro anglický jazyk. Více o jazykové podpo e v kapitole 9.

1.6 Správa struktury dokumentu



Pravá ást hlavního okna programu zobrazuje vlastní text dokumentu. Levá ást zobrazuje strukturu dokumentu, viz obr. 1.3.

V tomto návodu budeme ozna ovat strukturální ásti dokumentu, jako jsou ásti, kapitoly, nebo sekce, souhrnným názvem *sekce dokumentu*. Levá horní ást hlavního okna zobrazuje panel globální struktury dokumentu, zobrazující celkovou strukturu dokumentu, tedy všechny jeho sekce, hierarchickým zp sobem. Tento panel m žeme použít pro navigaci dokumentem, nebo pro zm -nu struktury dokumentu. Pro vytvo ení nové sekce dokumentu klikneme pravým tlačítkem myši na nadazenou sekci, nap . ást, a zvolíme p íkaz Vytvo it *sekci* z nabídky, kde *sekce* ozna uje typ konkrétní sekce, tedy nap . ást, kapitola apod. Smazání sekce provedeme výb rem sekce pomocí myši a stiskem klávesy Delete. Sekce lze p esouvat jejich tažením myši.

Panel v pravé ásti hlavního okna zobrazuje tu ást dokumentu, která je vybraná v panelu globální struktury dokumentu. Program vždy rozd luje dlouhé dokumenty, jako jsou knihy, na editovatelné ásti na úrovni kapitol. Panel editace tedy v každém okamžiku zobrazuje práv jednu kapitolu knihy. V p ípad že p ípravujeme lánek, je zobrazen celý dokument najednou.



Obr. 1.3 Hlavní okno programu BookEditor

Panel lokální struktury dokumentu, zobrazený v levé dolní části hlavního okna, detailně zobrazuje strukturu právě upravované části dokumentu. Můžeme vybrat požadovanou rozlišovací úroveň pomocí tlačítek na liště:  pro úroveň sekcí,  pro úroveň odstavců, nebo **ab** pro úroveň slov. Tento panel můžeme rovněž použít jak pro navigaci, tak pro změnu struktury dokumentu. Objekty lze přesouvat tažením myši, i mazat stiskem klávesy Delete. Více v sekci 2.2.

Kapitola 2

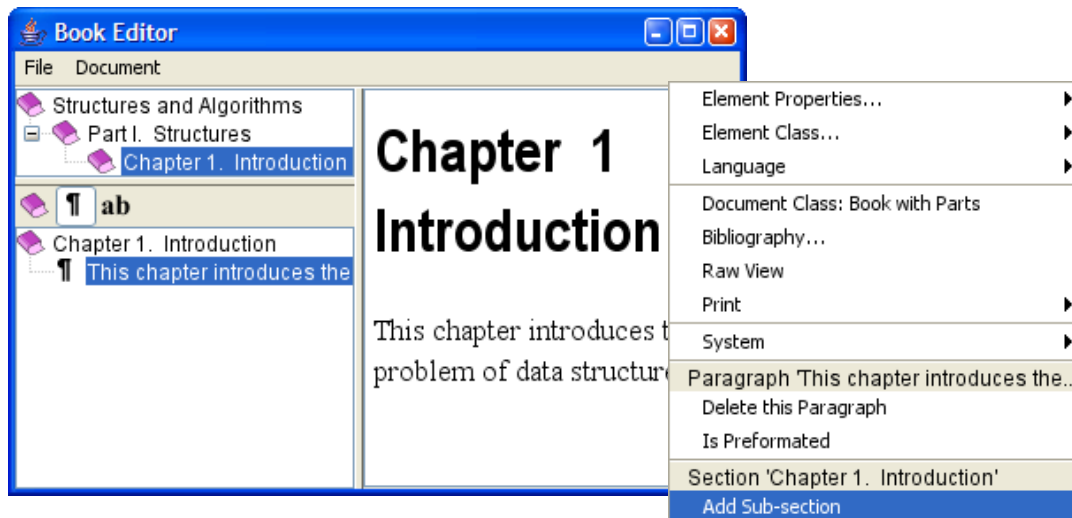
Základy editace

V této kapitole popíšeme základní postupy pro úpravu dokumentu.

2.1 Sekce dokument

V předchozí kapitole jsme popsali, jak vytvořit nový dokument, a jak přidat hlavní sekce (části a kapitoly). Nyní popíšeme, jak dále strukturovat kapitoly zobrazené v okně editoru.

Jak dále uvidíme, mnoho příkazů může být aktivováno pomocí kontextového menu. Pokud chceme například přidat novou sekci do kapitoly, stiskneme pravé tlačítko myši v místě, kde chceme novou sekci vytvořit. Zobrazí se kontextové menu podobné tomu na obr. 2.1.



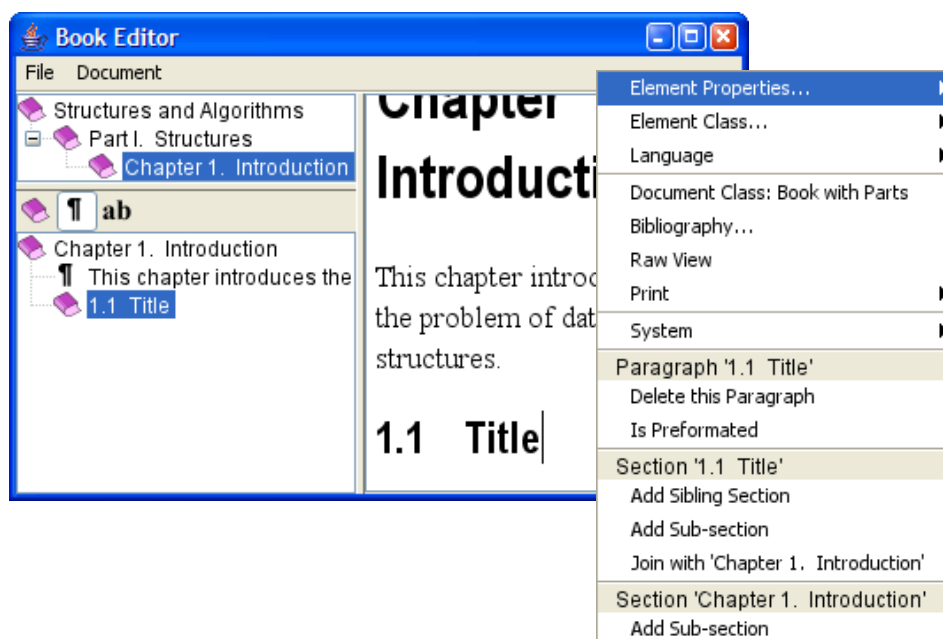
Obr. 2.1 Vytvoření nové sekce pomocí kontextového menu

V horní části menu vidíme příkazy aplikovatelné na celý dokument. Dále následují příkazy vztahující se k aktuální části dokumentu, od nejkonkrétnější – aktuální odstavec, k nadřazenější – aktuální kapitola. Pokud zvolíme poslední příkaz, do dokumentu je přidána nová sekce, viz obr. 2.2.

Pokud zobrazíme kontextové menu ještě jednou, aktuální části dokumentu jsou už tři: aktuální odstavec (titulek sekce), sekce 1.1, a kapitola 1. Nyní můžeme například vytvořit sekci na stejné úrovni jako sekci 1.1, tedy sekci 1.2, nebo její podsekci, tedy sekci 1.1.1. Také můžeme aktuální sekci spojit s nadřazenou sekcí. Strukturu dokumentu můžeme také měnit pouhým tažením objektů v Panelu lokální struktury dokumentu, zobrazeném v levé dolní části hlavního okna aplikace.

2.2 Přesouvání a kopírování objektů

Asto potebujeme přesouvat části dokumentu na jiné místo, nebo vytvářet kopie existujících částí. Editor nabízí dva způsoby přesunu a kopírování objektů. Buďto tradičním způsobem pomocí *šchránky*, nebo pomocí manipulací s objekty v Panelu lokální struktury dokumentu pomocí myši



Obr. 2.2 Nová sekce byla vytvořena


2.2.1 Kopírování a přesouvání pomocí schránky

Způsob Copy & Paste neboli kopírování a vložení je tradiční způsob kopírování a přesouvání objektů. Nejprve musíme označit materiál, který chceme zkopírovat, buďtažením kurzoru myši přes část dokumentu, nebo pomocí šipek na klávesnici a současným držetím klávesy Shift. Výběr je indikován změnou barvy textu a pozadí vybraného textu. Je-li výběr proveden, můžeme vybranou část buď zkopírovat do schránky, stiskem klávesy Ctrl-Insert, nebo ho vyjmout (zkopírovat do schránky, a zároveň vymazat výběr z dokumentu) stiskem Shift-Delete. Jakmile máme ve schránce nějaký obsah, můžeme ho vložit na zvolené místo stiskem Shift-Insert. Tímto způsobem můžeme vytvořit tolik kopií kolik potřebujeme.

2.2.2 Kopírování a přesouvání pomocí panelu struktury dokumentu


Alternativou k použití schránky je možnost manipulace s objekty v panelu struktury dokumentu. Tato metoda je vhodnější, pokud chceme přesouvat nebo kopírovat celé objekty jako jsou odstavce, tabulky, sekce apod. Přesunutí objektu lze provést tažením reprezentace objektu v panelu struktury dokumentu pomocí myši. Objekt můžeme přesunout vedle existujícího objektu, což je indikováno vodorovnou čarou nad nebo pod cílovým objektem. Případně lze objekt přesunout do existujícího objektu, například sekce, což je indikováno obdélníkem okolo cílového objektu.

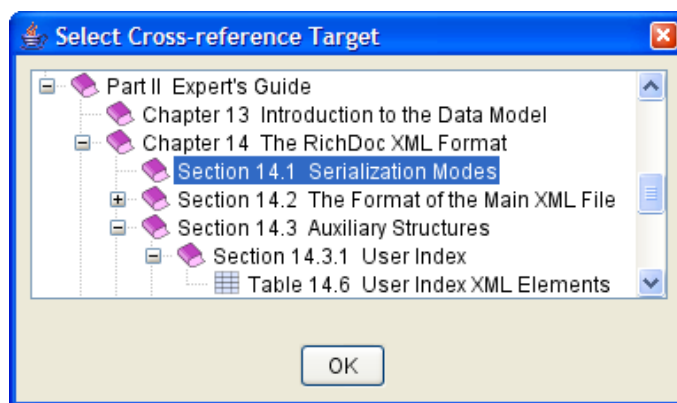
2.3 Formátování odstavce

Při psaní odstavce lze na text aplikovat různé formátovací styly. To lze provést příslušnými tlačítky na panelu tlačítek: **B** zapíná tučný text, *I* kurzívu, **T** strojopis, U podtržený text, a  přeškrtnutý text. Lze také použít odpovídající klávesové zkratky Ctrl+B, Ctrl+I, Ctrl+T, Ctrl+U a Ctrl+Shift+S. Pokud jste vybrali nějaký text, formátování je aplikováno na tento vybraný text. Pokud žádný výběr neexistuje, ale textový kurzor je uprostřed nějakého slova, formátování je aplikováno na toto slovo. Jinak není formátování aplikováno na žádný existující text, ale na text, který teprve bude napsán. Pokud aplikujeme formátovací atribut na text, který již tento atribut nese, je příslušný atribut vypnut.

Existují další speciální atributy, které lze aplikovat na text. Tlačítkem **EN** lze nastavit jazyk určité části textu, pokud je odlišný od jazyka zbytku textu. To může být užitečné pro zajištění správné funkce jazykových závislých funkcí, jako je kontrola pravopisu nebo dělení slov. Tlačítko **IDX** vytvoří heslo rejstříku, a vytvoří asociaci vybraného textu s tímto slovem, viz též sekci 6.1. Tlačítko **X** zruší formátovací atributy ve vybraném textu.

2.4 Křížové odkazy

Pro vložení křížového odkazu, nebo také hypertextového odkazu, na slovanou část dokumentu (sekci, tabulku, obrázek, rovnici apod.) použijte tlačítko , nebo stiskněte klávesy **Ctrl-L**. Zobrazí se okno Výběr cíle křížového odkazu, viz obr. 2.3.




Obr. 2.3 Okno Výběr cíle křížového odkazu

Po výběru cílového objektu, buď dvojitým kliknutím myši nebo stiskem klávesy **Enter**, je do dokumentu vložen křížový odkaz, ukazující na vybraný objekt. Odkaz zobrazuje text popisující cílový objekt, například „kapitola 5” nebo „(3.2)”. Odkaz je propojen s cílovým objektem, takže pokud se slovaní cílového objektu změní, text odkazu je automaticky opraven.

Křížový odkaz můžete také použít pro navigaci k cílovému objektu, kliknutím myši na odkaz. Pokud se nacházíte v editačním režimu, je třeba před stiskem podržet klávesu **Ctrl**, jinak je odkaz pouze vybrán pro editaci.

Pokud nám automaticky vytvořený text tvořící odkaz nevyhovuje, můžeme odkaz vybrat, a z kontextového menu vybrat příkaz Editovatelný. Po aplikaci příkazu lze text odkazu editovat, tedy změnit na libovolný text. Tento text však už nebude opraven, pokud se slovaní cílového objektu změní.





Všimněte si, že text odkazu obsahuje, kromě čísla cílového objektu, také popis typu objektu, například „kapitola 5”. To je vhodné zejména v případě, kdy je námi vytvořený dokument používán v různých kontextech, a typ objektu tedy v době přípravy dokumentu není znám. V závislosti na kontextu ve kterém je dokument použit to může být například část, kapitola nebo sekce. Proto je lepší nevyplnit pevně daný název, ale nechat program automaticky doplnit až v době, kdy je kontext znám. V češtině je však někdy potřeba text skloovat, například „v kapitole 5”. V takovém případě můžeme buď vypnout volbu Přidat název cíle, čímž zajistíme, že křížový odkaz bude obsahovat pouze číslo cílového objektu, ale nikoli jeho jméno. Pak můžeme jméno, které již není součástí textu odkazu, před odkaz dopsat ručně. Druhá možnost je pomocí menu Morfologie vybrat vhodný mluvnický tvar jména, například 4. pád.

Do dokumentu můžeme také vložit hypertextový odkaz na jakoukoli externí stránku na Internetu definovanou její URL adresou. Pokud začneme psát adresu stránky, po vložení začátku adresy, například `http://`, se psaný text automaticky změní na hypertextový odkaz, indikovaný zelenou barvou. Tento způsob je vhodný, pokud text odkazu je stejný jako jeho URL adresa. Pokud tomu tak není, můžeme napsat nejprve text odkazu, tento text vybrat, a stisknout tlačítko . Text se přemění na hypertextový

odkaz, a jsme dotázáni na příslušnou URL adresu. Adresa může být později změněna na použití příkazu Nastav URL kontextového menu.



2.5 Setting Properties of Objects







2.6 Seznamy











Editor nabízí tři typy seznamů: neuspořádané seznamy, uspořádané seznamy, popisné seznamy a TO-DO seznamy. Pro vložení prázdného seznamu určitého typu použijte příslušné tlačítko panelu tlačítek: , ,  nebo . Neuspořádaný seznam označuje své prvky pomocí různých značek. Uspořádaný seznam používá různé typy číslování (arabské číslice, římské číslice, písmena latinská nebo řecké abecedy, apod.) Položka popisného seznamu obsahuje hlavičku, sázenou tučnou, následovanou její definicí. TO-DO seznamy označují položky buď prázdným políčkem, označujícím nedokončenou položku, nebo zaškrtnutým políčkem, označujícím dokončenou položku. Stav položky lze změnit kliknutím na její políčko.

Je-li kurzor na konci položky seznamu, lze za tuto položku přidat novou položku stiskem klávesy Enter. Rovněž lze použít kontextové menu: můžeme přidat prázdnou položku před nebo za aktuální položku.


2.7 Tabulky

Novou tabulku přidáme do dokumentu stiskem tlačítka  nebo . První příkaz vloží tabulku s úlovaným titulkem, druhý příkaz vloží pouze tabulku. Pomocí druhé volby lze tabulky do sebe libovolně ovádat. Vytvořená tabulka má implicitně jedinou buňku. Pokud chceme vytvořit tabulku obsahující více buněk, stiskneme tlačítko na panelu tlačítek, a tažením myši nastavíme požadovanou velikost.

Pro změnu struktury tabulky použijeme příslušný panel tlačítek. Tlačítka  a  použijeme pro vložení nového sloupce na pravý okraj tabulky, respektive nového řádku na spodek tabulky. Podobná tlačítka  a  použijeme pro vložení sloupce nebo řádku před nebo nad aktuální buňku. Tlačítka  a  slouží pro vymazání aktuálního sloupce nebo řádku.

Panel tlačítek také obsahuje příkazy pro manipulaci s aktuální buňkou. Tlačítka  a  slouží pro sloužení aktuální buňky s jejím pravým nebo spodním sousedem. Text v buňce může být zarovnán doleva () , centrován () , doprava () , podle desetinné tečky () nebo může být vyrovnán (). Podobně lze zarovnávat vertikálně, a to nahoru () , doprostřed () nebo dolů ().

U tabulek lze definovat vlastnosti jejich rámečků. Implicitně tabulky nemají viditelný rámeček. V editačním režimu však rámeček je i tak zobrazen, ale pouze jako vizuální pomůcka. Rámeček tabulky lze definovat pomocí azením tabulky do *řady*, viz sekce 2.5. Například `border` definuje standardní rámeček tabulky.

Jednotlivé části rámečků tabulky lze předefinovat. Můžete zvolit jeden z předefinovaných typů rámečků, nebo nedefinovat vámi požadované vlastnosti rámečků. Změna rámečků se provede stiskem tlačítka  , a “obkreslením” požadované části rámečků pomocí myši. Požadovaný typ rámečků lze vybrat ze seznamu zobrazeného vedle tlačítka pro kreslení rámečků.

Kapitola 3

Editor rovnic

RichDoc Framework disponuje vestavěným editorem rovnic a matematických výrazů. Pomocí tlačítka \sqrt{x} vložíme do dokumentu *inline* rovnici, tedy rovnici nepřerušující tok textu v odstavci, například $e^{ix} + 1 = 0$. Druhý typ rovnice je tzv. *zobrazená rovnice* vložená pomocí $\overline{\sqrt{x}}$, která je umístěna na samostatný řádek, a je oddělena od okolních objektů vertikální mezerou, například

$$\int \frac{1}{1+x^2} dx = \tan^{-1} x$$

Více informací o zobrazených rovnicích naleznete v sekci 3.7.

3.1 Matematický text

Základem obsahu rovnic je matematický text. Text může označovat proměnné, konstanty, čísla, funkce apod. Matematický text rozdělujeme do několika kategorií popsaných v tabulce 3.1. Každá kategorie má přiřazen určitý typografický styl, který může být v případě potřeby změněn. O vizuálních stylech detailně pojednává kapitola 5.

Tabulka 3.1 Typy matematického textu

Typ	Popis	Styl	Příklad
mtext	generický text	normální	text
var	proměnná	kurzíva	<i>x</i>
const	konstanta	normální	<i>e</i>
num	číslo	normální	10
dim	rozměr	normální	kg
vec	vektor	tučně	a
mat	matice	tučně	A
dom	obor (například celá čísla)	zdvojené	\mathbb{Z}
mathSf	bezpatkové	bezpatkové	A

Pokud vám žádná uvedená kategorie nevyhovuje, můžete použít kategorii generický text, a vytvořit její podkategorii pomocí systému vizuálních stylů a též, jak je popsáno v kapitole 5.

Začnete-li psát text, typ textu se nastaví automaticky. Je-li první napsaný znak písmeno, nastaví se typ *proměnná*. Napíšete-li číselný výraz, nastaví se typ *číslo*. Ostatní typy textu je třeba nastavit ručně, výběrem ze seznamu zobrazeného na panelu tlačítek.

Můžete také vložit do rovnice běžný text, pomocí tlačítka π . Na tento text lze aplikovat veškeré vlastnosti jako na běžný odstavec, jako formátování, vkládání inline grafiky, nebo dokonce inline rovnice. Do odstavce však nelze vložit řádkový zlom.

3.2 Operátory a symboly

Do matematických vzorců můžete vkládat různé typy operátorů a symbolů. Skupiny tlačítek **a** a Γ slouží pro vkládání znaků malé resp. velké řecké abecedy. řecká písmena jsou považována za nor-

mální matematický text. Malá ecká písmena jsou implicitn typu *prom nná*, zatímco velká písmena jsou typu *rozm r*. Skupiny tla ítek \neq a ∞ slouží pro vložení matematického operátoru nebo symbolu.

Uživatelé zb hlí v systémech TeX/LaTeX mohou vkládat operátory i symboly p ímo napsáním odpovídajícího LaTeXového p íkazu. Nap íklad po napsání ‘\equiv’ je text automaticky nahrazen odpovídajícím operátorem \equiv . Tabulky 3.2, 3.3, 3.4 a 3.5 shrnují všechny podporované operátory, symboly a ecká písmena.

Tabulka 3.5 zobrazuje pouze speciální operátory které nemají odpovídající klávesu na b žné klávesnici. Operátory, které takovou klávesu mají, jako jsou ‘<’, ‘=’, ‘+’, etc., stejn jako interpunk - ní symboly jako ‘,’ ‘;’, ‘!’ lze samoz ejm vložít prostým stiskem odpovídající klávesy.

Tabulka 3.2 Písmena malé ecké abecedy

Písmeno	Zkratka	Písmeno	Zkratka
α	<code>\alpha</code>	β	<code>\beta</code>
γ	<code>\gamma</code>	δ	<code>\delta</code>
ϵ	<code>\epsilon</code>	ζ	<code>\zeta</code>
η	<code>\eta</code>	θ	<code>\theta</code>
ι	<code>\iota</code>	κ	<code>\kappa</code>
λ	<code>\lambda</code>	μ	<code>\mu</code>
ν	<code>\nu</code>	ξ	<code>\xi</code>
π	<code>\pi</code>	ρ	<code>\rho</code>
ς	<code>\varsigma</code>	σ	<code>\sigma</code>
τ	<code>\tau</code>	υ	<code>\upsilon</code>
φ	<code>\varphi</code>	χ	<code>\chi</code>
ψ	<code>\psi</code>	ω	<code>\omega</code>

Tabulka 3.3 Písmena velké ecké abecedy

Písmeno	Zkratka	Písmeno	Zkratka
Γ	<code>\Gamma</code>	Δ	<code>\Delta</code>
Θ	<code>\Theta</code>	Λ	<code>\Lambda</code>
Ξ	<code>\Xi</code>	Π	<code>\Pi</code>
Σ	<code>\Sigma</code>	Φ	<code>\Phi</code>
\Chi	<code>\Chi</code>	Ψ	<code>\Psi</code>
Ω	<code>\Omega</code>		

3.3 Struktury

Vzorce mohou obsahovat krom jednoduchých struktur jako jsou prom nné, operátory a symboly, také složit jší struktury jako jsou zlomky, odmocniny, integrály apod. Tyto struktury mohou být vloženy stiskem p íslušného tla ítku na matematické nástrojové lišt , nebo napsáním jejich LaTeXového ekvivalentu. Po vložení struktury se zobrazí n kolik obdélník , které slouží jako výchozí body pro vkládání materiálu do struktur. Mezi t mito body se m žete pohybovat pomocí kurzorových tla ítek. Struktury lze do sebe samoz ejm libovoln vno ovat. Typy struktur najdete v tabulce 3.6.

Tabulka 3.4 Symboly

Symbol	Zkratka	Symbol	Zkratka
∞	<code>\infty</code>	∂	<code>\partial</code>
ϑ	<code>\vartheta</code>	ϱ	<code>\varrho</code>
ϕ	<code>\phi</code>	\Re	<code>\Re</code>
\Im	<code>\Im</code>	$'$	<code>\prime</code>
■	<code>\blacksquare</code>	□	<code>\Box</code>
∇	<code>\nabla</code>	\backslash	<code>\backslash</code>
ℓ	<code>\ell</code>	$\sqrt{\quad}$	<code>\sqrt{\quad}</code>
\mathcal{L}	<code>\schwellL</code>	©	<code>\copyright</code>
\leftarrow	<code>\leftarrow</code>		

Tabulka 3.5 Operátory

Operátor	Zkratka	Operátor	Zkratka
\geq	<code>\ge</code>	\leq	<code>\le</code>
\gg	<code>\gg</code>	\ll	<code>\ll</code>
\geqslant	<code>\geqslant</code>	\leqslant	<code>\leqslant</code>
\neq	<code>\ne</code>	\times	<code>\times</code>
\pm	<code>\pm</code>	\mp	<code>\mp</code>
\cup	<code>\cup</code>	\cap	<code>\cap</code>
\vee	<code>\vee</code>	\wedge	<code>\wedge</code>
\forall	<code>\forall</code>	\exists	<code>\exists</code>
\in	<code>\in</code>	\approx	<code>\approx</code>
\simeq	<code>\simeq</code>	\sim	<code>\sim</code>
\cdot	<code>\cdot</code>	\equiv	<code>\equiv</code>
\propto	<code>\propto</code>	\perp	<code>\perp</code>
\odot	<code>\odot</code>	\oplus	<code>\oplus</code>
\leftarrow	<code>\leftarrow</code>	\leftarrow	<code>\leftarrow</code>
\rightarrow	<code>\rightarrow</code>	\leftrightarrow	<code>\leftrightarrow</code>
\rightarrow	<code>\rightarrow</code>	\Leftarrow	<code>\Leftarrow</code>
\Rightarrow	<code>\Rightarrow</code>	\Leftrightarrow	<code>\Leftrightarrow</code>
\dots	<code>\dots</code>	\vdots	<code>\vdots</code>
\ddots	<code>\ddots</code>	$\cdot\cdot\cdot$	<code>\cdot\cdot\cdot</code>
\cdots	<code>\cdots</code>	\circ	<code>\circ</code>
•	<code>\bullet</code>	\star	<code>\star</code>
\rightsquigarrow	<code>\rightsquigarrow</code>		

Vzhled n kterých struktur závisí na tom, zda jsou *zobrazené* i nikoli. Jsou-li v zobrazené, jsou v tší a zabírají více vertikálního místa. Porovnejte $\frac{x}{x+1}$ a $\frac{x}{x+1}$, nebo $\int_0^1 x dx$ a $\int_0^1 x dx$. Zobrazený stav je implicitn nastaven, je-li struktura v zobrazené, nikoli inline rovnici. Vno ování struktur rov-

Tabulka 3.6 Matematické struktury

	Zkratka	Popis	Příklad
x_1^2	<code>_ or ^</code>	index	$x_1, 2^n$
$\frac{a}{b}$	<code>\frac</code>	zlomek	$\frac{a+b}{a-b}$
$\sqrt[3]{x}$	<code>\sqrt</code>	odmocnina	$\sqrt[3]{x+1}$
$\sum x$	<code>\sum</code>	suma	$\sum_{i=1}^n x_i$
$\prod x$	<code>\prod</code>	produkt	$\prod_{i=1}^n x_i$
$\int x$	<code>\int</code>	integrál	$\int_0^1 x^2 dx$
$\oint x$	<code>\oint</code>	kruhový integrál	$\oint_0^1 x^2 dx$
$\iint x$	<code>\iint</code>	dvojný integrál	$\iint_A x dA$
$\iiint x$	<code>\iiint</code>	trojný integrál	$\iiint_A x dA$
$\iiiiiint x$	<code>\iiiiiint</code>	tverný integrál	$\iiiiiint_A x dA$
$\int \dots \int x$	<code>\idotsint</code>	skupina integrál	$\int \dots \int_A x dA$

není zobrazený stav, tedy struktura vnořená v jiné strukturu je implicitně není zobrazená. Chceme-li změnit zobrazený stav manuálně, použijeme tlačítko Zobrazen v nástrojové liště struktury. Je-li tlačítko nezaškrtnuté () , není struktura zobrazena, je-li zaškrtnuté () , struktura zobrazena je, a je-li v nerozhodnutém stavu () , má struktura implicitně zobrazený stav, v závislosti na jejím umístění.

Další stavová proměnná ovlivňující vzhled struktury je *limity*, která označuje, zda budou například limity struktury zobrazeny nad a pod touto strukturou, například $\int_a^b x$, nebo vpravo od struktury, například $\int_a^b x$.

3.4 Závorky

Vzorce můžeme doplnit o různé typy závorek, viz tabulka 3.7. Prázdný pár závorek lze vložit buď pomocí skupiny tlačítek $\langle x \rangle$, nebo napsáním příslušné klávesové zkratky. Při napsání příkazu pro levou závorku se automaticky doplní odpovídající pravá závorka.

Tabulka 3.7 Závorky

	Zkratka	Popis	Příklad
$\langle x \rangle$	<code>(</code>	kulaté závorky	$\langle x \rangle$
$[x]$	<code>[</code>	hranaté závorky	$[x]$
$\{x\}$	<code>{</code>	složené závorky	$\{x\}$
$ x $	<code> </code>	rovné závorky	$ x $
$\langle x \rangle$	<code>\langle</code>	úhlové závorky	$\langle x \rangle$
$\lfloor x \rfloor$	<code>\lfloor</code>	zaokrouhlení dolů	$\lfloor x \rfloor$
$\lceil x \rceil$	<code>\lceil</code>	zaokrouhlení nahoru	$\lceil x \rceil$


Je-li kurzor uprostřed páru závorek, zobrazí se v nástrojovém okně lišta vlastností závorek. Zde můžeme nastavit typ levé a pravé závorky samostatně. Tak lze vytvořit nesymetrické závorky, jako například $(0, 1)$.

Velikost závorek se implicitně nastavuje automaticky, podle materiálu obklopeného závorkami. Velikost závorek však můžete nastavit ručně, zvolením hodnoty Velikost na nástrojové liště. Tak můžete vytvořit například takovou rovnici: $((x+1)(x+2))^2$. Za normálních okolností by vnější závorky měly stejnou velikost jako závorky vnitřní.

3.5 Pole


Pole jsou jedno- nebo dvoudimenzionální struktury podobné tabulkám. Proto se vytváření polí podobá tvorbě tabulek, popsané v sekci 2.7. Pole využijeme pro tvorbu matic a vektorů, a jenom pro vizuální strukturování vzorců. Můžeme snadno vytvářet rovnice jako například tato:

$$\begin{pmatrix} 1 & 3 & -1 \\ 2 & 1 & 3 \\ -5 & 8 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 7 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Nové pole vložíme buď pomocí tlačítka , nebo napsáním příkazu `\array`. Zadáváme-li pole pomocí tlačítka, lze po jeho stisknutí a před jeho uvolněním pomocí myši nastavit požadované rozměry pole. Změna struktury pole se provádí stejně jako u tabulek, viz sekce 2.7. Podobné je i nastavování rámečků polí.

U jednotlivých buněk pole lze nastavit jejich horizontální zarovnání, stejně jako u buněk tabulky. Vertikální zarovnání však nastavit nelze, vertikální zarovnání buněk se vždy řídí pravidly matematické typografie. Buněk ve stejném řádku jsou tedy zarovnány na jejich matematickou osu, která probíhá v tštinou vertikálním středem obsahu buněk.

3.6 Mezery

Editor rovnic nastavuje mezery mezi prvky rovnice automaticky, v souladu se zvyklostmi matematické sazby. V některých případech však potřebujeme toto implicitní chování změnit, tedy rozšířit nebo zúžit existující mezery. Explicitní mezeru vložíme do rovnice pomocí skupiny tlačítek . Zelená barva označuje pozitivní mezery, způsobí tedy rozšíření existující mezery. Negativní mezery jsou označeny červeně. Stejně jako u tabulek a polí, můžeme vložit několik mezer daného typu najednou, stiskem tlačítka myši a jejím tažením doprava. Typy mezer můžeme vidět v tabulce 3.8. Velikosti mezer jsou relativní vzhledem k velikosti použitého písma. Například nejširší mezerka, `\quad`, je stejně široká jako výška znakové bučky použitého fontu.

Tabulka 3.8 Horizontální mezery

Šířka	Tlačítko	Zkratka	Příklad	Šířka	Tlačítko	Zkratka	Příklad
1/16		<code>\,</code> nebo <code>\thinspace</code>	$\Rightarrow\Leftarrow$	-1/16		<code>\negthinspace</code>	$\Rightarrow\Leftarrow$
1/8		<code>\:</code> nebo <code>\medspace</code>	$\Rightarrow\Leftarrow$	-1/8		<code>\! nebo \negmedspace</code>	$\Rightarrow\Leftarrow$
1/4		<code>\;</code> nebo <code>\thickspace</code>	$\Rightarrow\Leftarrow$	-1/4		<code>\negthickspace</code>	$\Rightarrow\Leftarrow$
1		<code>\quad</code>	$\Rightarrow\Leftarrow$				

3.7 Zobrazené rovnice

Tato sekce popisuje vlastnosti, které jsou specifické pro zobrazené rovnice, tedy rovnice umístěné na samostatném řádku.

Zobrazené rovnice mohou být íslované. íslovaný i ne íslovaný stav rovnice p epneme pomocí p íkazu íslovat kontextového menu. M žeme také vytvo it rovnici která je implicitn íslovaná, použijeme-li tlačítko

íslované rovnice mohou být rozd leny do n kolika ádk . ádky rovnic vkládáme a rušíme stejn jako ádky tabulek, tedy pomocí tlačítek (p idat ádek), (vložit ádek) a (zrušit ádek), tentokrát však z nástrojové lišty Zobrazená rovnice. Je-li zobrazená rovnice íslovaná, m žeme zvolit bu íslování po ádcích, nebo p id lit celé rovnici jediné íslo. Stav íslování zm níme volbou íslovat po ádcích z kontextového menu. íslujeme-li po ádcích, lze u každého ádku zvláš nastavit íslovaný i ne íslovaný stav.

ádky rovnice lze rozd lit na zarovnané skupiny pomocí *tabulátor* . Tabulátory jsou dvou druh : zarovnávají bu to doprava (tlačítko), nebo doleva (tlačítko). Aby tabulátory správn pracovaly, je třeba vložit do každého ádku rovnice alespo jeden tabulátor. Tabulátory se pak zarovnají do stejné horizontální pozice. M žeme použít více tabulátor , všechny ádky by však m ly obsahovat stejný počet tabulátor . Následující rovnice byla vytvo ena pomocí doprava zarovnávacích tabulátor , umíst ných p ed rovnítky.

$$\begin{aligned}x + 1 &= 10 \\x - y + 2 &= 2 \\y &= 6\end{aligned}$$

Stejného efektu by bylo možné docílit i pomocí polí, viz sekce 3.5. Tabulátory jsou však vhodn ější pro zadávání nezávislých rovnic, které jsou pouze vzájemn zarovnané. U polí také není možné p i a zovat jednotlivým ádk m ísla.

Krom ádk p edstavující rovnice lze vkládat ádky obsahující text, pomocí tlačítek resp. , pro p idávání nebo resp. vkládání odstavcových ádk . To se m že hodit pro vložení textu do rovnice aniž by byla rovnice p erušena, z d vodu zachování íslování nebo zarovnaní. Nap íklad m žeme vytvo it takovouto rovnici:

$$\begin{aligned}A_1 &= N_0(\lambda; \Omega) - \phi(\lambda; \Omega'), \\A_2 &= \phi(\lambda; \Omega') - \phi(\lambda; \Omega),\end{aligned}$$

a

$$A_\infty = \mathcal{N}(\lambda; \omega).$$

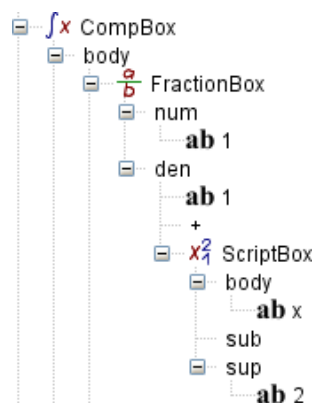
Všimn me si, že rovnice je správn zarovnaná, což by nebylo možné bez vložení textu *dovnit* rovnice.

3.8 Úprava rovnic

Editor rovnic umož ůuje p esouvat a kopírovat objekty, jak je popsáno v sekci 2.2. M žeme použít ob metody, tedy použít schránku nebo panel struktury dokumentu. Panel však musíme p epnout do nejpodrobn ějšího režimu, pomocí tlačítka **ab**. Panel m žeme také využít pro p ehledné zobrazení struktury složitě rovnic. Nap íklad struktura níže uvedeně rovnic je zobrazena na obr. 3.1.

$$\int \frac{1}{1+x^2} dx = \tan^{-1} x$$

P i p íprav složitých rovnic m že editor rovnic zobrazít velké množství obdél níků sloužících pro dopln ní materiálu. N kdy se m že zdát, že jich je více než je pot eba. Nap íklad na obr. 3.2 vlevo jsou dva obdél níky hned vedle sebe. P estože to není na první pohled z ejmé, je mezi nimi rozdíl: první obdél ník leží *vn* argumentu druhé mocniny, zatímco druhý leží *uvnit* tohoto argumentu. Editor rovnic tyto dv pozice rozlišuje, p estože nemusí mít vliv na kone ný vzhled rovnice. V tomto konkrétním p ípad bychom m li vložit materiál do prvního obdél níku, tedy *vn* argumentu mocniny, jelikož p ípadn ěná rovnice by vložený materiál takto interpretoval. M li bychom te-



Obr. 3.1 Hierarchický pohled na část rovnice

dy vždy dodržovat zvyklosti pro matematickou sazbu, a používat vizuální pomůcky jako jsou závorky, protože editor rovnic dokáže explicitně zachytit priority operací, které však nemusí být viditelné v její finální vizuální prezentaci rovnice.



Obr. 3.2 Příklad výchozích editačních bodů složitěho výrazu

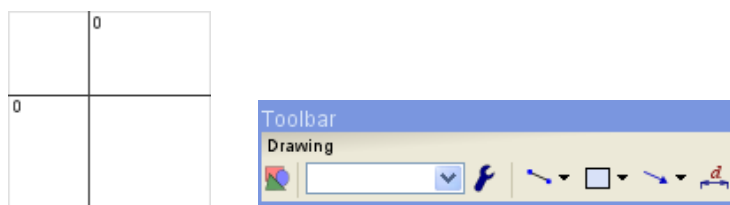
Kapitola 4

Kreslení obrázk

Pomocí editoru můžeme upravovat jednoduché obrázky, obsahující 2D vektorovou nebo rastrovou grafiku. Obrázky mohou obsahovat text, rovné i zakřivené tvary, obdélníky, spojky, šipky a rastrové obrázky.

4.1 Nový obrázek




Nový obrázek vložíme do dokumentu pomocí tlačítka  nebo . Podobně jako u tabulek lze využít variantu bez nebo s číslováním titulkem. Po vložení obrázku se zobrazí prázdná kreslicí plocha, a nástrojová lišta Kreslení, viz obr. 4.1. Pomocí lišty přidáváme do obrázku nové grafické objekty.



Obr. 4.1 Prázdná kreslicí plocha a lišta Kreslení

4.2 Přidáváme grafické objekty

Přidání nového objektu provedeme výběrem *kreslicího nástroje* z lišty Kreslení. Jednotlivé nástroje jsou podrobně popsány v následujících sekcích.





Po výběru nástroje se zobrazí lišta Nový objekt, viz obr. 4.2. Pomocí této lišty můžeme nastavit požadované vlastnosti objektu, *dříve než* ho přidáme na kreslicí plochu. Tímto vlastnostmi můžeme být barva výplně () , barva popředí () , i další specifické vlastnosti (). Vlastnosti lze samozřejmě změnit i u objektu, které již byly do nástrojové lišty přidány.



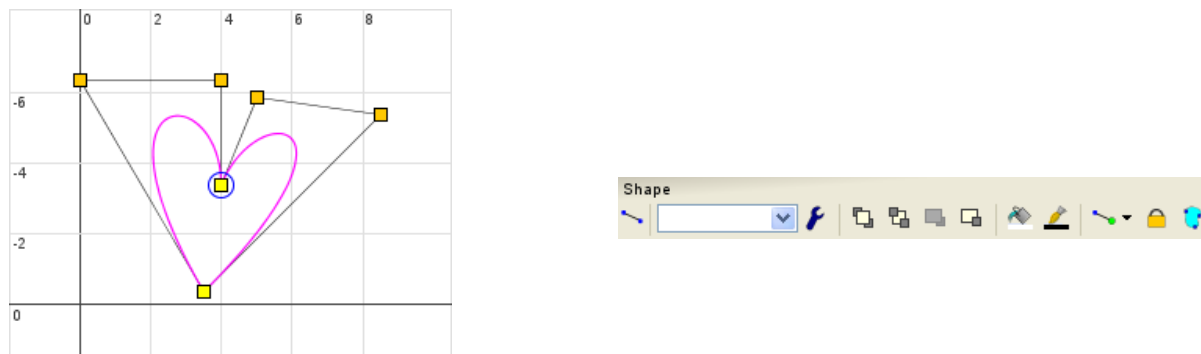
Obr. 4.2 Nástrojová lišta Nový objekt

První ikona na liště indikuje typ kreslicího nástroje, tedy typ objektu, který bude přidán do obrázku. Žlutá hvězdička napovídá, že se jedná o vlastnosti zatím neexistujícího, prototypového objektu, který bude teprve vytvořen. Pokud přesuneme kurzor myši nad kreslicí plochu, kurzor myši se změní do tvaru aktivního kreslicího nástroje. Jsme-li s nástrojem hotovi a chceme manipulovat s existujícími objekty na obrázku, zrušíme nástroj stisknutím klávesy ESC nebo pravého tlačítka myši.







4.2.1 Tvary



Řečné tvary jako jsou čáry, kvadratické křivky, Bézierovy křivky nebo oblouky, přidáme pomocí odpovídajících nástrojů , ,  nebo . Po zvolení nástroje nakreslíme zvolenou křivku tažením myši. Tímto nástrojem vytváříme tvary obsahující jediný segment.

Pro modifikaci tvaru zrušíme aktivní kreslicí nástroj, a vybereme tvar pomocí myši. Zobrazí se nástrojová lišta Tvar. Dále se zobrazí množství obdélníků podél tvaru, viz obr. 4.3. Tyto obdélníky reprezentují *manipulační body*, které můžeme použít pro změnu tvaru jejich tažením myši.



Obr. 4.3 Manipulační body segmentu tvaru







Pro přidání různých typů segmentů k označenému segmentu použijeme některý z nástrojů lišty Tvar: , , , nebo . Poté klikáním myši přidáváme segmenty zvoleného typu k označenému segmentu. Všimněte si rozdílu mezi nástrojem  a : první nástroj vytvoří nový tvar obsahující jediný segment, kdežto druhý nástroj přidá segment k označenému tvaru. Pro zrušení jednoho segmentu z nějakého tvaru klikneme na segment pravým tlačítkem myši a z kontextového menu vybereme příkaz Zruš Segment.

Tvary jsou implicitně *otevřené*, mají tedy začátek a konec. Pokud tvar *uzavřeme* příkazem  začátek a konec tvaru se propojí úsečkou. Opětovné použití příkazu zruší uzavřený stav. Uzavřené tvary mohou být *vyplněny* barvou. Vyplněný/nevyplněný stav přepneme tlačítkem .


4.2.2 Obdélníky

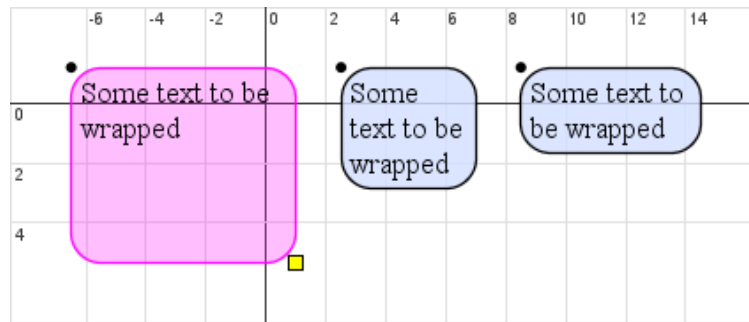
Do obrázku můžeme přidat uzavřené tvary obdélníkového i jiného tvaru. Tyto tvary mohou volitelně obsahovat text. Obdélník vytvoříme pomocí jednoho z nástrojů uvedených v tabulce 4.1. Po zvolení nástroje nakreslíme obdélník stiskem tlačítka myši a tažením doprava a dolů. Pomocí tlačítka **A** nastavíme zda obdélník obsahuje i neobsahuje text.

Tabulka 4.1 Nástroje pro tvorbu obdélníků

	Bez textu	S textem
Obdélníkový		
Se zakulacenými rohy		
Eliptický		

Jako text můžeme do obdélníku vložit jakýkoli materiál, například odstavec, sekvenci odstavců, seznam apod. Požadovanou šířku obdélníku nastavíme pomocí jeho žlutého manipulačního bodu, zobrazeného v jeho pravém dolním rohu, viz obr. 4.4. Text obdélníku je pak nalámán tak, aby se vešel do požadované šířky. Požadovaná výška může být větší než je potřeba pro vložení textu, část obdélníku je potom prázdná. Je-li však požadovaná výška příliš malá, je automaticky upravena tak, aby se text do obdélníku celý vešel.

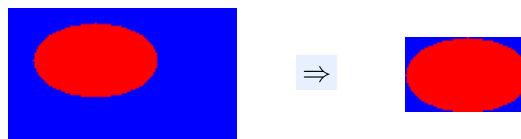
Obdélník obsahující tabulku vytvoříme pomocí nástroje . Opět můžeme pomocí žlutého bodu nastavit preferovanou velikost tabulky. Struktura tabulky se definuje stejným způsobem jako struktura běžné tabulky, viz sekce 2.7.



Obr. 4.4 Zlom textu v obdélíku

4.2.3 Rastrová grafika





Rastrový obrázek, také znám jako *bitmapa*, vložíme zkopírováním obrázku do schránky a vložením stiskem kláves Shift + Insert. Pomocí kontextového menu nastavíme vlastnosti obrázku. Píkazem Nastavit měřítko změněme obrázek zvětšit nebo zmenšit. Píkazem Okraje obrázku stejné barvy, viz obr. 4.5. Chceme-li, aby jedna z barev použitých v obrázku byla prhledná, použijeme píkaz Nastav prhlednou barvu, a myší označíme barvu která se má zprhlednit.



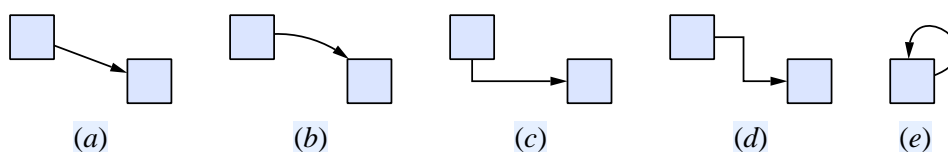
Obr. 4.5 Ozání rastrového obrázku

4.2.4 Spojky

Obdélíky a uzavřené tvary měžeme propojovat různými typy *spojek*. Spojka je rovná, lomená nebo zakřivená propojující dva objekty. Koncové body spojky se automaticky upraví při každé změně polohy nebo velikosti propojených objektů. Konce spojky mohou mít různou zakončení jako jsou šipky nebo kroužky.

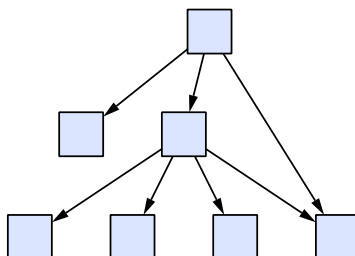
Novou spojku vytvoříme pomocí nástrojů  pro rovnou spojku,  pro zakřivenou spojku,  pro lomenou spojku, nebo  pro dvakrát zalomenou spojku. Spojku nakreslíme tažením myši. Pokud pohybuje kurzorem nad nějakým objektem, konec spojky se zafixuje na střed tohoto objektu. Při uvolnění tlačítka myši zůstane konec spojky spojen s tímto objektem. Pokud chceme zabránit fixování spojky na objekt, podržíme během tažení konce spojky klávesu Ctrl.

Je-li konec spojky zafixován na střed nějakého objektu, je tvar spojky oříznut podle okraje tohoto objektu, takže spojka vizuálně navazuje na okraj objektu, nikoli na jeho střed, viz obr. 4.6. Zakřivená spojka může mít identický začátek a konec, čímž vznikne *smyka*, viz obr. 4.6e. V tomto případě má spojka o jeden manipulační bod navíc, kterým nastavíme úhel natočení smyčky kolem jejich koncových bodů.




Obr. 4.6 Různé tvary spojky: (a) přímá spojka, (b) zakřivená spojka, (c) zalomená spojka, (d) dvakrát zalomená spojka, (e) smyka

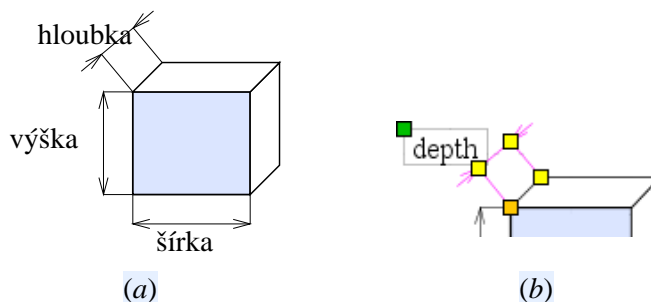
Pokud vytvoříme složitý diagram obsahující množství vzájemně propojených obdélníků, můžeme obdélníky automaticky rozmístit. Provedeme to vybráním kořenového obdélníku, a aktivací příkazu Automatické rozvržení z kontextového menu, viz obr. 4.7. Od této chvíle je poloha všech obdélníků s výjimkou kořenového řízena automaticky. V případě potřeby můžeme automatické rozvržení zase vypnout a ručně doladit rozmístění obdélníků.



Obr. 4.7 Automatické rozmístění obdélníků.



4.2.5 Lineární Kóty

Lineární kótu přidáme do obrázku pomocí nástroje , viz obr. 4.8a. Tvar kóty nastavíme pomocí příkazu Manipulací bodů, viz obr. 4.8b. Dva body určují polohu šipky kóty, další dva polohu koncových úseček. Poslední, zelený bod určuje pozici popisku kóty. Podobně jako u spojky můžeme kolmé úsečky zafixovat na části jiných objektů, a tím zajistit, že tvar kóty se automaticky upraví při změně geometrie kótovaného objektu.



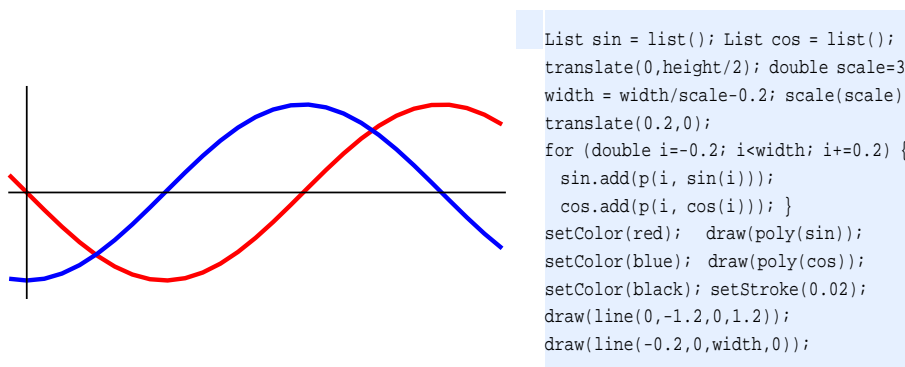
Obr. 4.8 Přídání kót: (a) Typy kót, (b) Manipulační body kóty

4.2.6 Skriptovaná grafika

Někdy je vhodné popsat část obrázku pomocí algoritmu, nebo skriptu, namísto interaktivního přidávání grafických objektů. K tomuto účelu slouží nástroj Skriptovaná grafika (). Po aktivaci nástroje tažením myši nadefinujeme základní obrys skriptovaného obrázku, a příkazem  otevřeme editor skriptu, kde naprogramujeme vzhled obrázku. Na obr. 4.9 vidíme příklad skriptu a jemu odpovídající grafiky. Podrobný popis skriptovacího jazyka je nad rámec této dokumentace.

4.3 Editace obrázků

Existující grafické objekty v obrázku můžeme jednoduše editovat. Posunutí objektu provedeme prostým tažením myši. Pokud má vybraný objekt manipulační body, je třeba před tažením kliknout mimo manipulační bod, aby došlo k posunu objektu, nikoli změně jeho tvaru. Potřebujeme-li posouvat více objektů najednou, provedeme výběr kliknutím mimo objekty, a tažením myši ohraničíme objekty, které chceme vybrat. Posunutí výběru poté provedeme tažením libovolného vybraného objektu. Z výběru







Obr. 4.9 Příklad skriptované grafiky

ru m žeme také p idat nebo odebrat jednotlivé objekty kliknutím p i stisknuté klávese Ctrl. Vybrané objekty smažeme stiskem klávesy Delete.

Velikost kreslicí oblasti nelze explicitn nastavit; její velikost je nastavena automaticky tak, aby pojala všechny vložené objekty. Pokud p esuneme objekt za okraj kreslicí oblasti, objekt do asn zmizí, ale po jeho umíst ní se kreslicí oblast zv tší tak aby se do ní p esunutý objekt vešel. Relativní poloha objekt v kreslicí oblasti tedy nemá význam. Pokud p esuneme *všechny* objekty v oblasti, vrátí se vzhled oblasti po p epo ítání jejich rozm r do p vodního stavu.

Objekty lze kopírovat dv ma zp soby: bu p esuneme objekty tažením myši a sou asn držíme stisknutou klávesu Ctrl, nebo zkopírujeme ozna ené objekty do schránky stiskem Ctrl + Insert, a stiskem Shift + Insert vložíme kopii obsahu schránky do obrázku. Druhý zp sob lze použít i pro kopírování objekt mezi obrázky. Je také možné vytvo it kopii celého obrázku pomocí panelu struktury dokumentu, viz sekce 2.2.

Objekty jsou p idány do obrázku v ur itém po adí, které ovliv uje jejich vzájemné p ekryvání. Objekt vložený pozd ji je umíst n “p ed” objektem umíst ným d íve. Pokud se takové objekty p ekryvají, pozd ji nakreslený objekt zakryje objekt nakreslený d íve. Vzájemné po adí objekt lze však dodatn upravovat. Objekt posuneme o pozici dop edu stiskem tla ítka  nebo dozadu stiskem tla ítka . Tla ítka  p enese objekt p ed všechny objekty, zatímco tla ítka  ho p enese za všechny objekty.

4.4 Setting Visual Properties

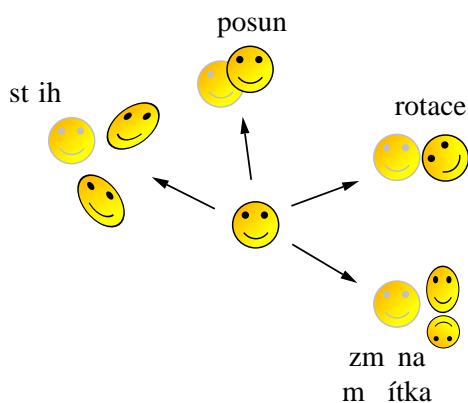
4.5 M ení textu

Jelikož Editor není p ísn typu WYSIWYG, jak bylo vysv tleno v sekci 1.1, mohou být p i nasazení dokumentu v jiném kontextu zm nny vlastnosti písma použitého v obrázku. To m že zm nit rozm - ry textu a tím poškodit vzhled obrázku. Proto se nedoporu uje pomocí styl m nit typ nebo velikost písma použitého v obrázcích.


Editor navíc implicitn p izp sobuje zobrazený text tak, aby byl dob e íitelný i na za ízeních s nízkým rozlišením, jako jsou obrazovky po íta . V rámci tohoto p izp sobení jsou nastaveny rozm - ry každého znaku tak, aby m l celo íselné rozm ry, což však zp sobuje, že p i zm n m ítka obrázku se rozm ry textu nem ní rovnom rn . Pokud nap íklad zmenšíme znak široký 11 bod na polovinu, bude široký 5 nebo 6 bod , což je zna ná odchylka od správné hodnoty 5,5 bodu. Ur ování rozm r textu je tedy zna n nep esné, s chybou ádu až desítek procent, což m že v d sledku op t zp sobit poškození struktury obrázku. Pokud náš obrázek vyžaduje p esné m ení velikosti textu, nastavíme vlastnost P esné m ení. Tím vynutíme rovnom rné zm nny rozm r textu p i zm n m ítka, ale zp sobíme pon kud nepravidelný vzhled textu zobrazeného v nízkém rozlišení.

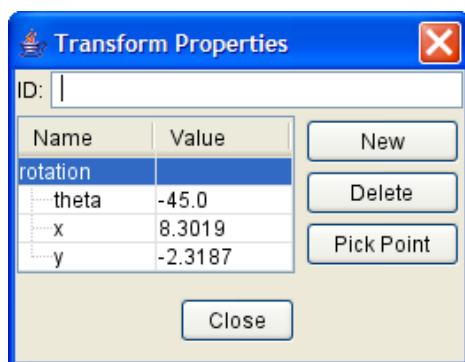
4.6 Transformace

Existující objekty lze transformovat pomocí transformačních operací. Tato vlastnost je zvláště užitečná ve spojení s animací, popsané v sekci 4.7. Tyto základní transformační operace jsou zobrazeny na obr. 4.10.



Obr. 4.10 Transformační operace

Novou transformaci přidáme do obrázku použitím tlačítka  na lištu obrázku. Zobrazí se okno Vlastnosti transformace, viz obr. 4.10. Zpočátku je seznam operací prázdný. Novou operaci přidáme stiskem tlačítka Nová, a výběrem jedné z těchto operací z nabídky. Do seznamu je přidána vytvořená operace spolu s odpovídajícími parametry. Různé operace mají různé parametry, viz tabulka 4.2.







Obr. 4.11 Okno Vlastnosti transformace

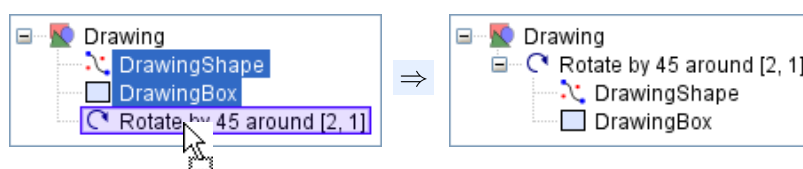
Každá transformace může mít identifikátor, který zadáme v okně Vlastnosti transformace. Tento identifikátor můžeme využít později, například v programové změně parametrů transformace za účelem dosažení efektu animace, viz sekce 4.7.

Do jedné transformace lze přidat více transformačních operací. Operace se v takovém případě aplikují na objekty postupně, jedna po druhé. Pro operace mající střed, tedy všechny operace kromě posunu, lze zadat střed transformace stiskem tlačítka Vyber bod a výběrem bodu v obrázku pomocí myši. Proměnné x, y vybrané operace jsou potom automaticky nastaveny podle vybraného bodu.

Je-li transformace definována, potom budeme zaříditi, aby byla aplikována na vybrané objekty. Aplikaci lze provést přesunem zvolených objektů do objektu transformace pomocí panelu lokální struktury dokumentu, viz obr. 4.12. Po přesunutí objektů se za ně transformace aplikovat na vybrané objekty. Transformační objekty lze do sebe takto vnořovat, a dosáhnout tak efektu skládání transformací.

Tabulka 4.2 Parametry transformačních Operací

Operace	Popis	Parametry
 Posun	Posune objekt o daný vektor	$x, y \dots$ vektor posunu
 Rotace	Otočí objekt o daný úhel okolo daného bodu	$x, y \dots$ střed otáčení, $theta \dots$ úhel otáčení ve stupních (ve směru hodinových ručiček)
 Změna měřítka	Změní měřítka objektu	$x, y \dots$ střed změny měřítka, $s_x \dots$ horizontální zvětšení, $s_y \dots$ vertikální zvětšení; lze použít záporné hodnoty pro převrácení objektu
 Stíhání	Deformuje objekt stíháním	$x, y \dots$ střed deformace, $s_x \dots$ horizontální stíhání, $s_y \dots$ vertikální stíhání



Obr. 4.12 Přesun objekt do transformace

4.7 Animace

Pomocí editoru obrázků můžeme snadno proměnit statické obrázky na obrázky animované, doplněním animačních pravidel. Animační pravidla pro obrázek definujeme výběrem obrázku a použitím příkazu Animační pravidla z kontextového menu, Zobrazí se okno Animační pravidla, viz obr. 4.13.



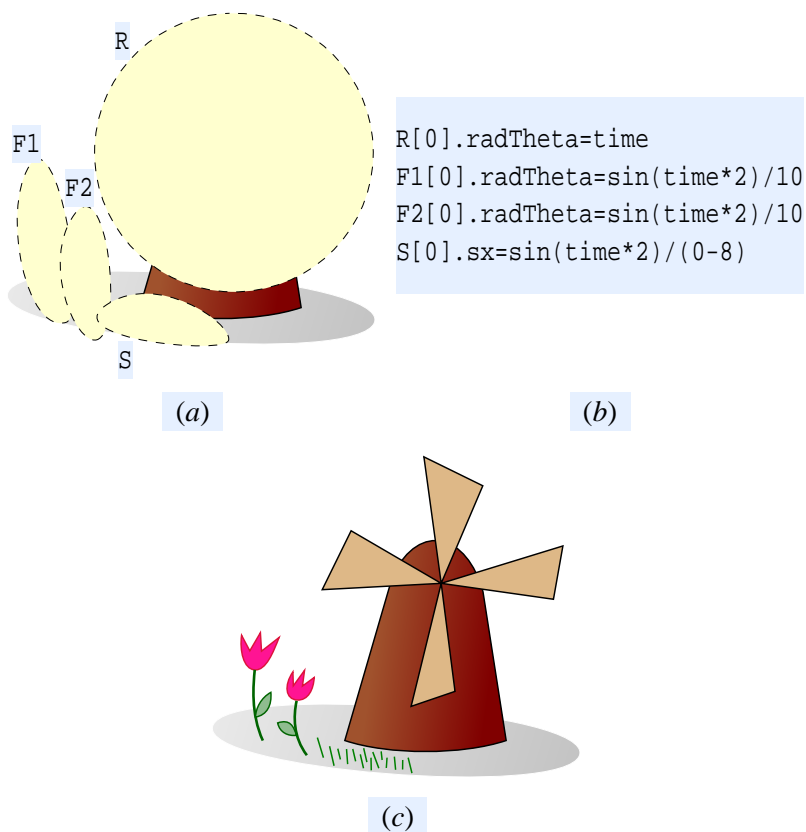
Obr. 4.13 Okno Animační pravidla

Každé pravidlo má tvar $objekt = hodnota$, kde $objekt$ je výraz který se vyhodnotí na objekt přijímající hodnotu, a $hodnota$ je výraz který se vyhodnotí na objekt který poskytuje hodnotu. Během animace jsou pravidla periodicky vykonávána, výrazy na pravých stranách jsou vyhodnocovány a přiřazovány do objektů na levých stranách pravidel. Objekty, které lze použít na levé straně pravidla jsou shrnuty v tabulce 4.3. Pravá strana pravidla může být aritmetický výraz, obsahující matematické operátory, funkce, a animační parametr $time$ (čas). Tento parametr reprezentuje reálný čas v sekundách, startující od nuly v okamžiku startu animace, tedy v okamžiku zobrazení prvního snímku animovaného obrázku.

Příklad animace je zobrazen na obr. 4.14. obr. 4.14a ukazuje transformace definované na obrázku. obr. 4.14b ukazuje animační pravidla. Na obr. 4.14c vidíme výsledný animovaný obrázek.

Tabulka 4.3 Animovatelné vlastnosti

Vlastnost	Popis
<i>transformace[n].parametr</i>	Parametr <i>n</i> -té operace <i>transformace</i> . Nabídka parametr závisí na typu operace, viz sekce 4.6.
<i>objekt.barva.složka</i>	Nastavuje barvu <i>objektu</i> . <i>barva</i> může být jedno z <i>foreColor</i> (barva pop edí), <i>backColor</i> (barva pozadí), <i>backColor2</i> (alternativní barva pozadí při gradientním vypl ování) nebo <i>borderColor</i> (barva ráme ku). <i>složka</i> je jedno z <i>r, g, b</i> pro specifikaci barvy v RGB modelu (červená-zelená-modrá), nebo jedno z <i>c, m, y</i> pro specifikaci barvy v CMY modelu (azurová-fialová-žlutá). Složky barev by m ly být v intervalu $\langle 0, 1 \rangle$.

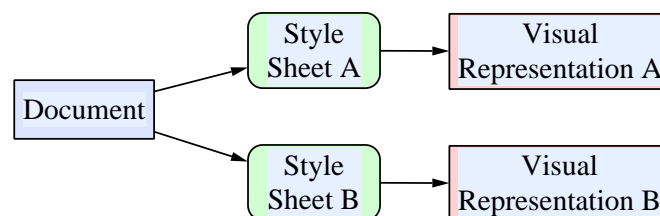


Obr. 4.14 Ukázka animovaného obrázku: (a) parametrizovaný obrázek, (b) anima ní pravidla, (c) animovaný obrázek

Kapitola 5

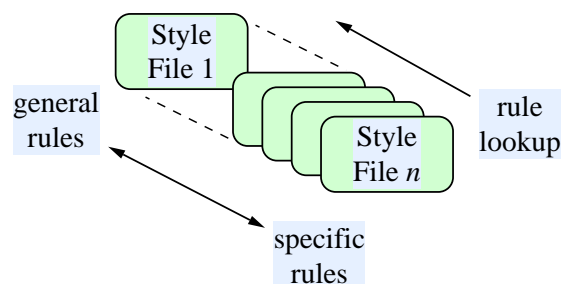
Managing Visual Styles

As we already mentioned, the key difference between the RichDoc framework and classical strictly WYSIWYG word processors is sharp separation of document structure from its visual representation. To get some specific visual representation of a document, we must apply some visual formatting rules to the document. These rules are defined in a structure called *style sheet*. Since the style sheet does not depend on the document, we may use different style sheets for the same document to get alternative visual representations of the document, see obr. 5.1. It is, of course, also possible to apply the same style sheet to a group of documents to provide uniform look and feel for such group.



Obr. 5.1 Applying Visual Rules to a Document

The style sheet structure may be split into several *cascades*, to provide rules of different levels of generality, see obr. 5.2. There may be general style files providing rules applicable for wide variety of situations, which may be refined with more specific style files providing exceptions from general cases. When the visual presentation module needs to find a specific visual formatting rule, it first checks the more specific style files, and continues up the cascade until an appropriate rule is found. The style language used to define style files is described in detail in kapitola 19.

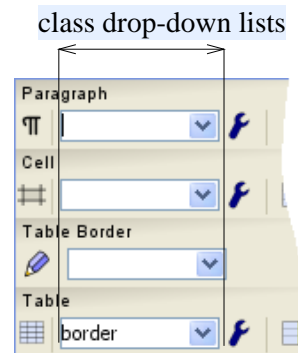


Obr. 5.2 Cascading Style Sheets

As described in sekce 2.5, it is possible to assign each document element some visual attributes such as font and color, in an ad-hoc manner. This is convenient for handling special cases, but for regularly occurring patterns, it is recommended to treat visual attributes more systematically.

For each document element type, such as paragraph, paragraph fragment, table or section, has some standard set of visual properties that is acquired from the style sheet currently in use. We may, however, declare special named variants, called *classes* for any element type, and override the visual properties for these. For some elements, there are already some predefined classes, such as class border for tables, but you can create your own classes. For each class, you may define a set of attributes in the same way as you define attributes for a concrete document elements. You can then *associate* instances of document elements with an appropriate class, which causes the element to *inherit* all vi-

sual attributes from its class. You can later change the attributes of the class, and all elements associated with that class immediately reset their appearance to reflect the changes. It is of course possible to override the inherited values with ad-hoc settings for each individual element.



Obr. 5.3 Specifying Class from the Toolbar

We can specify the name of element class using the toolbar, see obr. 5.3. Select the element for which you want to set the class, and access the drop-down list on its toolbar. Here either type the class name, or select it from the list. When the class is specified, the element view immediately acquires visual attributes from the class and resets its appearance accordingly.

Kapitola 6

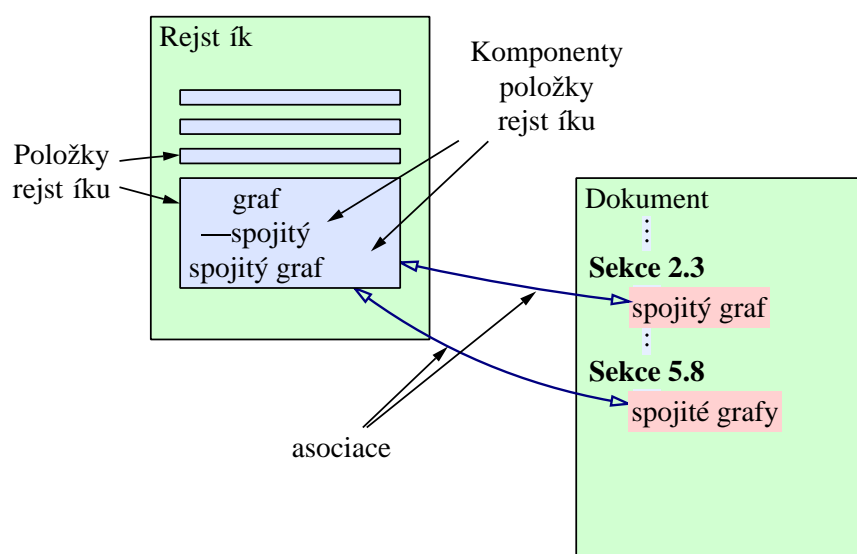
Rejstík, glosář a literatura

V této kapitole popíšeme přípravu rejstíku, glosáře a seznamu bibliografických citací.

6.1 Příprava rejstíku a glosáře

Kvalitní tištěné knihy obsahují na konci rejstíky nebo glosáře (*glossary*, slovníkové pojmy). Pomocí editorů můžeme tyto struktury snadno připravit. Rejstíky obvykle obsahují abecední seznam hesel, s odkazy na stránky, kde jsou hesla definována nebo zmíněna. Glosáře obsahují seznam hesel opatřených krátkou definicí.

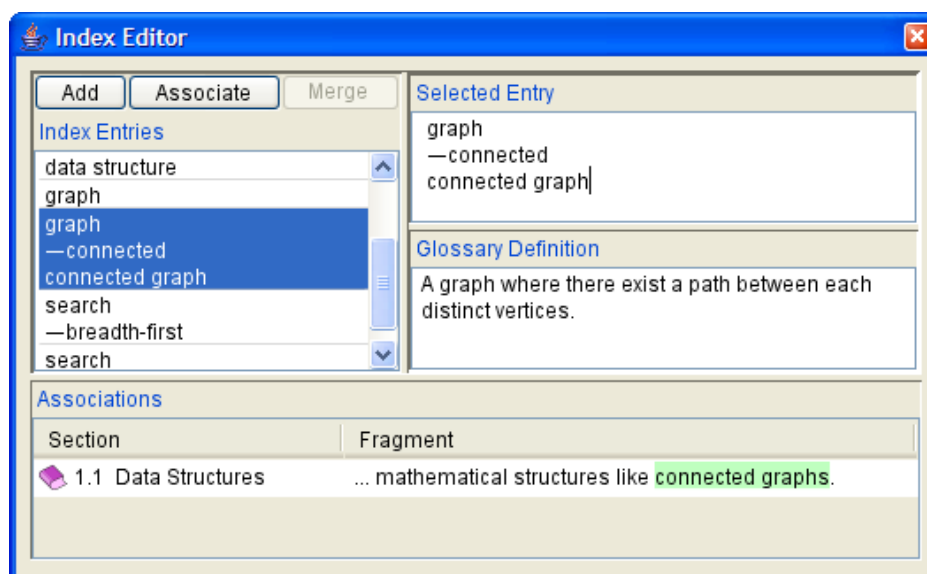
Pro vytvoření rejstíku musíme vytvořit seznam hesel, a tato hesla navázat na části textu obsažené v dokumentu. Definici jednoho hesla budeme nazývat *položka rejstíku*. Položka může být propojena s několika objekty nebo částmi textu v dokumentu, viz obr. 6.1. Položka také může být opatřena glosářovou definicí. Pro přípravu položek indexu zvolíme příkaz Rejstík... Zobrazí se okno Editor rejstíku, viz obr. 6.2.



Obr. 6.1 Struktura rejstíku

Okno obsahuje čtyři panely; panel Položky rejstíku zobrazuje abecední seznam všech definovaných položek, neboli hesel. V panelu Vybraná položka můžeme změnit text vybrané položky. Panel Glosářová definice umožňuje propojit vybranou položku její definicí. Panel Asociace zobrazuje seznam propojení vybrané položky rejstíku s částmi dokumentu.

Jak vidíme na obr. 6.2, heslo může být rozděleno do několika částí, vyjmenovaných od hlavního doplnkové, například *graf—spojitý*. Na které položky mohou mít shodnou hlavní část, a lišit se jen v doplňkových částech (například *prohledávání—do šířky*, *prohledávání—do hloubky*). Takové uspořádání ovlivní finální vzhled rejstíku obvyklým způsobem: společná část se objeví pouze jednou, a doplňkové části budou vyjmenovány pod společnou hlavní částí. V editoru rejstíku je však třeba u každé položky vyjmenovat všechny její části. Také si všimněte, že jedna položka rejstíku může definovat několik alternativních znění hesla, například *graf—spojitý* a *spojitý graf*. Taková položka se tedy objeví v rejstíku několikrát. Příklad konečného vzhledu rejstíku je zobrazen na obr. 6.3.



Obr. 6.2 Okno Editor rejstříku

Index

algorithm, 3, 5	graph
- A^* , 3	- connected, 3
connected graph, 3	search
data structures, 3, 4	- breadth-first, 3
	- depth-first, 3

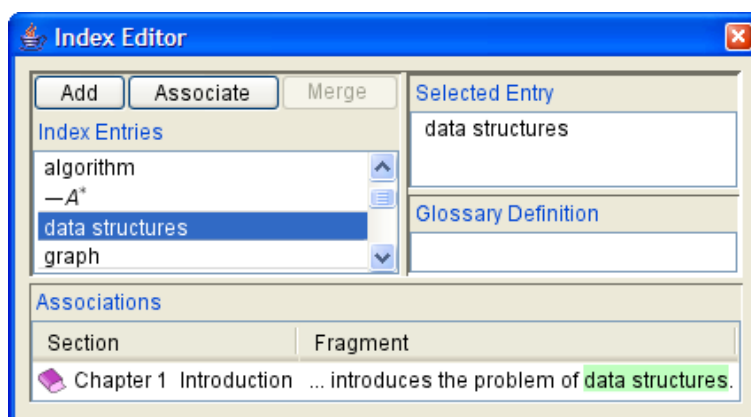
Obr. 6.3 Příklad konečného vzhledu sekce Rejstřík

nelu Vybraná položka. Chceme-li definovat více alternativních hesel pro jednu položku, jednoduše umístíme každé z nich na samostatný řádek. Chceme-li definovat heslo skládající se z n kolika částí, doplnkovou část uvedeme na další řádek, a použijeme pomlčku pro odsazení. Pro definici částí této úrovně odsadíme pomocí dvou pomlček, atd. Takto lze definovat libovolné množství úrovní, nedoporučují se však hesla rozdelená na více než tři části.

Náš příklad také demonstruje, že pro definici hesel můžeme použít nejen text, ale i formátování, inline rovnice i grafiku, tedy jakýkoli materiál, který lze použít v běžném odstavci.

Jakmile vytvoříme položky rejstříku, je třeba je propojit s obsahem dokumentu. Můžeme je propojit například se sekcemi i odstavci, ale nejlépe je propojíme s částmi odstavců, které přímo odpovídají citaci daného hesla. Položku rejstříku propojíme s částí dokumentu výběrem objektu nebo části textu v dokumentu, a stiskem tlačítka Propojit v editoru rejstříku. Pokud byl označen jaký text, bude položka propojena s tímto textem. V opačném případě bude položka propojena s objektem, ve kterém je umístěn kurzor. Je-li takové určení objektu nejednoznačné, je třeba vybrat konkrétní objekt ve zobrazeném menu. Vytvořené propojení se pak objeví v seznamu Asociace.

Položky indexu lze definovat i opačně: nejprve označíme nějaký text v dokumentu, a potom stiskneme tlačítko **IDX** v nástrojové liště Odstavec (pod tlačítkem **U**). Vybereme-li například v dokumentu text "data structures" a stiskneme **IDX**, zobrazí se Editor rejstříku podobný tomu na obr. 6.4. Vidíme, že do seznamu položek byla přidána položka odpovídající vybranému textu, která byla s tímto textem automaticky propojena, viz seznam Asociace. Pokud přesné znění vybraného textu není vhodné pro položku indexu, lze položku indexu opravit, na text dokumentu to nebude mít vliv.



Obr. 6.4 Položka indexu vytvo ená z vybraného textu

rovými klávesami a podržením klávesy Shift, nebo kliknutím a sou asným držením klávesy Ctrl. Poté stiskneme tlačítko Sloužit. Jednotlivá hesla vybraných položek budou sloužit, stejně jako asociace vybraných položek s textem dokumentu.

Asociaci položky zrušíme výběrem ze seznamu asociací, a stiskem klávesy Delete. Můžeme také zrušit celou položku včetně všech jejích asociací, výběrem položky v seznamu položek a stiskem Delete.

6.2 P íprava seznamu bibliografických citací

Pro vdecké, ale i jiné dokumenty, často potřebujeme na konec dokumentu připojit seznam citovaných zdrojů, na které se například odkazujeme z hlavní části dokumentu. Proces přípravy seznamu citací lze rozdělit na tyto části:

1. Vytvoření databáze citací, nezávislé na dokumentu.
2. Vybrání seznamu citací z databáze relevantních pro daný dokument.
3. Přidání odkazu do seznamu vytvořeného kroku 2. do hlavní části dokumentu.

6.2.1 P íprava databáze bibliografických záznamů

Prvním krokem při tvorbě seznamu citací je vytvoření databáze bibliografických záznamů. Jelikož se předpokládá, že jeden uživatel může použít jednu citaci v několika dokumentech, je tato databáze externí, tedy nezávislá na konkrétním dokumentu. Přípravu databáze zahájíme příkazem Literatura... z menu Nástroje. Zobrazí se seznam citací příslušejících k aktuálnímu dokumentu, který je zatím prázdný. Po stisknutí tlačítka Databáze se zobrazí okno editoru bibliografických záznamů, vše viz obr. 6.5.

V horní části okna vidíme hierarchický seznam bibliografických záznamů. Záznamy lze přidat do skupin. Novou skupinu vytvoříme tlačítkem Nová skupina. Nový záznam přidáme do vybrané skupiny tlačítkem Nový záznam. Pokud v seznamu vybereme nějaký záznam, zobrazí se jeho údaje v dolní části okna.

Bibliography Database

New Group New Item Add Item Dump

Database

- Proceedings
 - [i P] Intelligent user-support system for modeling and simulation
 - [pd] Proceedings of 2002 IEEE CCA/CACSD Conference

Selected Item

Field	Value
Type	Proceedings
Label	glasgowProc
Language	English
Title	Proceedings of 2002 IEEE CCA/CACSD Conference
Year	2001
Editor	
Publisher	
Organization	
Address	Glasgow
Month	
Note	
Key	

The year of publication or, for an unpublished work, the year it was written. This field's text should contain only numerals.

Close

Obr. 6.5 Editor bibliografických záznam

Nejprve je třeba zvolit typu záznamu ze seznamu uvedeného v tabulce 6.1. Po zvolení typu se pro příslušným způsobem upraví seznam ostatních údajů záznamu. Pole zobrazená tu jsou povinná: pokud nejsou vyplněna, je tato chyba indikována jejich červenou barvou. Pole zobrazená normálním písmem jsou nepovinná a mohou tedy zůstat nevyplněna. Seznam všech polí pro jednotlivé typy publikací je v tabulce 6.2. Povinná pole jsou zde opět zobrazena tučně.

Každý záznam by měl mít explicitně nastavený jazyk. Údaje o publikaci by měly být vyplněny stejným jazykem jako je jazyk publikace. Pokud plánujeme citaci publikace z dokumentu v jazyce odlišném od jazyka publikace, můžeme některá pole záznamu, například název, přeložit do potřebovaného jazyka. Pokud například chceme citovat českou publikaci v záznamu v angličtině, měly by být údaje o publikaci vyplněny česky, ale můžeme přidat anglický název publikace. Při citaci publikace se potom použije pole v jazyce odpovídající danému kontextu, pokud je k dispozici.

Pole Autor a Editor mohou mít více než jednu hodnotu, pokud má publikace více autorů nebo editorů. V takovém případě můžeme buď tažením myši vytvořit potřebovaný počet řádků pro příslušné pole, nebo pro zadání nové hodnoty jednoduše stisknout klávesku (', '). Nepotřebná pole mohou být vymazána jejich výběrem a stiskem Delete. V jednom řádku by však nikdy nemělo být zadáno více jak jedno jméno!

Tabulka 6.1 Types of Bibliographical References

Type	Description
[iP] In Proceedings	An article in the proceedings of a conference.
[pd] Proceedings	The proceedings of a conference.
[bk] Book	A book with an explicit publisher.
[iB] In Book	A part of a book, which may be a chapter and/or a range of pages.
[mt] Master's Thesis	A Master's thesis.
[pt] PhD Thesis	A PhD thesis.
[iC] In Collection	A part of a book with its own title.
[mn] Manual	Technical documentation.
[tr] Technical Report	A report published by a school or other institution, usually numbered within a series.
[bl] Booklet	A work that is printed and bound, but without a named publisher or sponsoring institution.
[jn] Journal	A journal or magazine series, without particular issue.
[ji] Journal Issue	A journal or magazine issue.
[at] Article	An article from a journal or magazine.
[up] Unpublished	A document with an author and title, but not formally published.
[ms] Misc	Use this type when nothing else seems appropriate.
[us] Unstructured	A citation with no structure, defined by a markup text. Please do not use. Provided for compatibility with existing unstructured citations.

Informace o jedné publikaci m ůže být rozprost ena do n kolika záznam ů. Nap íklad lánek publikovaný ve sborníku konference je popsán pomocí dvou záznam ů: Záznam *Ve sborníku* popisuje detaily lánku (název, autor apod.), zatímco záznam *Sborník* popisuje detaily samotného sborníku (editory, vydavatele, rok vydání). Pokud vytvo íme nový záznam typu *Ve sborníku*, poslední pole záznamu nese název 'V', kam je třeba vyplnit odkaz na nad azený záznam typu *Sborník*. Po kliknutí na toto pole se zobrazí dv ů možnosti: Nový..., pro vytvo ení nového záznamu pro sborník, a Prohlí ůet..., pro vybrání existujícího záznamu. Po nalezení nebo vytvo ení záznamu se p ídají údaje o nad azeném záznamu na konec seznamu, viz obr. 6.6.

Tabulka 6.2 Fields of Bibliographical Records

Name	Description	Applies To
Address	Publisher's address. For major publishing houses, just the city is given. For small publishers, you can help the reader by giving the complete address.	Proceedings, Book, In Book, Master's Thesis, PhD Thesis, Manual, Technical Report, Booklet, Unpublished
Author	The name of the author.	In Proceedings, Book, In Book, Master's Thesis, PhD Thesis, In Collection, Manual, Technical Report, Booklet, Article, Unpublished, Misc
Chapter	A chapter number.	In Book
Edition	The edition of a book - for example, "second".	Book, In Book, Manual
Editor	Name of editor. If there are also "author" field(s), then the "editor" field gives the editor of the book or collection in which the reference appears.	Proceedings, Book
How Published	How something strange has been published.	Booklet, Misc
Institution	The institution that published the work.	Technical Report
Key	Used for alphabetizing and creating a label when the "author" and "editor" fields are missing. This field should not be confused with the Label field.	<i>all</i>
Language	Language of the publication.	<i>all</i>
Month	The month in which the work was published or, for an unpublished work, in which it was written.	Proceedings, Book, In Book, Master's Thesis, PhD Thesis, Manual, Technical Report, Booklet, Journal Issue, Misc
Note	Any additional information that can help the reader.	<i>all</i>
Number	The number of a journal, magazine, or technical report. An issue of a journal or magazine is usually identified by its volume and number; the organization that issues a technical report usually gives it a number.	Technical Report, Journal Issue
Organization	The organization sponsoring a conference.	Proceedings, Manual
Pages	A page number or range of numbers such as "42--111"; you may also have several of these, separating them with commas: "7,41,73--97".	In Proceedings, In Book, In Collection, Article
Publisher	The publisher's name.	Proceedings, Book, In Book
School	The name of the school where a thesis was written.	Master's Thesis, PhD Thesis
Series	The name of a series or set of books. When citing an entire book, the "title" field gives its title and an optional "series" field gives the name of a series in which the book is published.	Book, In Book
Text	Unstructured description of the citation.	Unstructured
Title	The work's title.	In Proceedings, Proceedings, Book, In Book, Master's Thesis, PhD Thesis, In Collection, Manual, Technical Report, Booklet, Journal, Article, Unpublished, Misc
Type	The type of a technical report - for example, "Research Note".	Technical Report
Volume	The volume of a journal or multivolume book work.	Book, In Book, Journal Issue
Year	The year of publication or, for an unpublished work, the year it was written. This field's text should contain only numerals.	Proceedings, Book, In Book, Master's Thesis, PhD Thesis, In Collection, Manual, Technical Report, Booklet, Journal Issue, Unpublished, Misc

Field	Value
Type	In Proceedings
Label	glasgow
Language	English
Author	Michal Ševčenko
Author	Heřman Mann
Title	Intelligent user-support system for modeling and simulation
Pages	
Note	
Key	
In	Proceedings
[pd] Language	English
[pd] Title	Proceedings of 2002 IEEE CCA/CACSD Conference
[pd] Year	2001
[pd] Editor	
[pd] Publisher	
[pd] Organization	
[pd] Address	Glasgow
[pd] Month	
[pd] Note	
[pd] Key	

Obr. 6.6 et zení bibliografických záznam

6.2.2 Kompatibilita se systémem BibTeX

Jak jste si možná všimli, systém organizace bibliografických záznam je podobný systému BibTeX. Editor disponuje nástrojem pro import záznam ve formátu BibTeXu (soubor .bib). Nástroj automaticky provádí potřebné transformace, jako je delení vícehodnotových polí na samostatná pole, i delení složených záznam na samostatné z et zené záznamy. Nejsou však podporovány všechny funkce BibTeXu, například textová makra.

6.2.3 O databázi

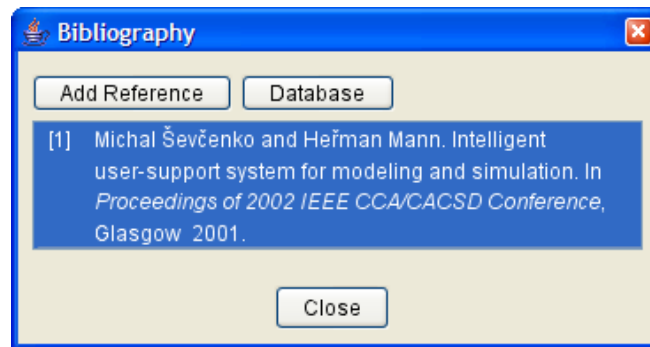
Databáze bibliografických záznam je určena pro osobní použití. Je uložena v souboru `~/ksmsa/bibliography/.database`, kde `~` označuje uživatelský adresář, ve Windows obvykle umístěný v adresáři `c:\Documents and Settings\username`. Pokud chceme tuto databázi používat z více počítačů, je třeba tento soubor ručně zkopírovat. Víceuživatelský systém pro správu bibliografických záznam zatím není implementován.

Formát databáze se může měnit mezi jednotlivými verzemi RichDoc frameworku. Chcete-li instalovat novější verzi Frameworku, je třeba databázi vyexportovat do souboru ve formátu XML, a po instalaci naimportovat zpět.

6.2.4 Odkazy na bibliografické záznamy

Po naplnění databáze údaji je možné přidávat citace do seznamů citací jednotlivých dokumentů. Pro přidání vybraných záznamů do aktuálního dokumentu použijeme tlačítko **Přidat záznam**. Okno zobrazující databázi se zavěsí, a vybrané záznamy jsou přidány do seznamu citací aktuálního dokumentu, viz obr. 6.7.

Nyní můžeme do hlavní části dokumentu přidávat odkazy na seznam citací. Odkaz přidáme stiskem tlačítka **Přidat odkaz**, nebo dvojitým kliknutím na položku v seznamu citací. Do dokumentu se



Obr. 6.7 Seznam citací konkrétního dokumentu

na pozici kurzoru přidá text reprezentující odkaz na citaci, například “[10]”. Tento text je automaticky aktualizován pokud se po adí citace v seznamu změní.

Kapitola 7

Version Management and Localization

This chapter discusses how the RichDoc Framework supports version management of a document, in sekce 7.1. sekce 7.2 describes how the RichDoc Framework supports the management of localized versions of a master document.

7.1 Version Management

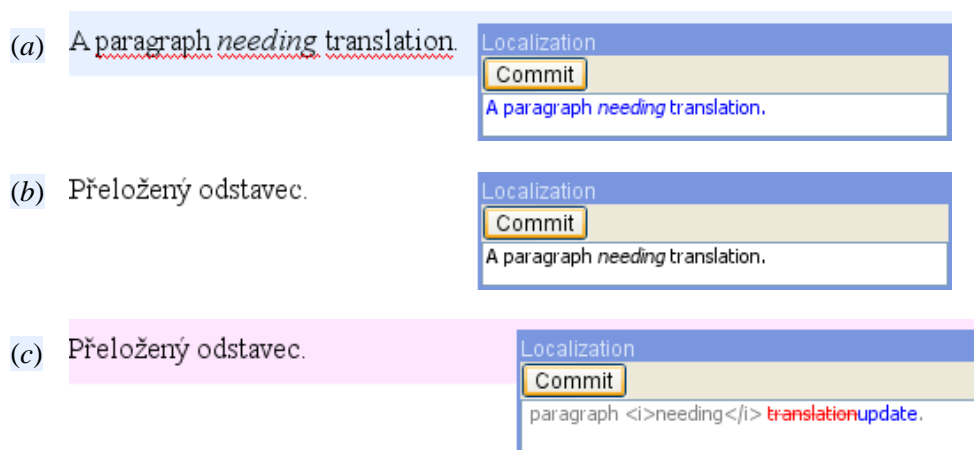
Under Construction

7.2 Managing of Localized Documents

Under Construction

Often you need to write a document, and provide localized versions of that document. It is also common that the master document is updated several times in its lifetime, and you would like to update the localized subversions accordingly. The RichDoc Framework's Localized Versions Management (LVM) greatly facilitates these tasks.

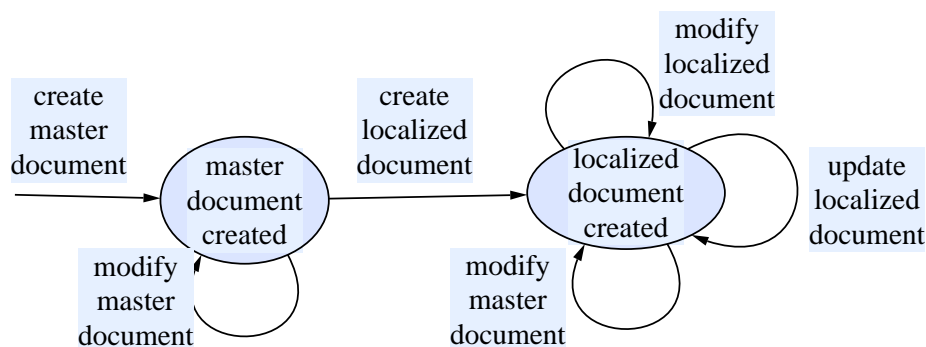
When you finish authoring of certain version of your document, and you want to create a localized version of it, use Tools → Create Localized Document command. A new document is created, which is initially a mere copy of the master document. When you open the newly created document, all paragraphs are displayed with blue background, indicating that they need translation. When you place text cursor into that paragraph, a localization window shows up. The localization window displays the difference between two master versions that needs to be accommodated into the localized document, see obr. 7.1a.



Obr. 7.1 States of Localization Paragraphs: (a) Created, (b) Up-to-date, (c) Modified

Initially, the whole master paragraph is displayed in blue color, indicating that new material has been added and needs to be translated. When you finish your translation, press the Commit button to confirm that the translation corresponds to the current version of the master document. The correspondence is indicated by normal white background of the paragraph and normal black text color in localization window, see obr. 7.1b. When you later modify the master document and update the localized document, all paragraphs in the localized document needing revision are displayed with purple bac-

kground. If you put the text cursor into that paragraph, the localization window shows the change that has been done in the master since last translation, see obr. 7.1c. When you finish the revision of the translation, press the Commit button to confirm that the paragraph corresponds to the current version of the master again.



Obr. 7.2 Localized Document Life Cycle

The life cycle of the documents is summarized in obr. 7.2. First, you must create the master document, and develop it for some time. When you create the localized document, you have two independent documents that can be modified independently, perhaps by different persons. When you finish some version of the master document, you may update the localized document by invoking the Create Localized Slave again, and then revise the localized document by modifying localized paragraphs that have been modified in the master document. Of course, you may create and manage many localized documents for different languages.

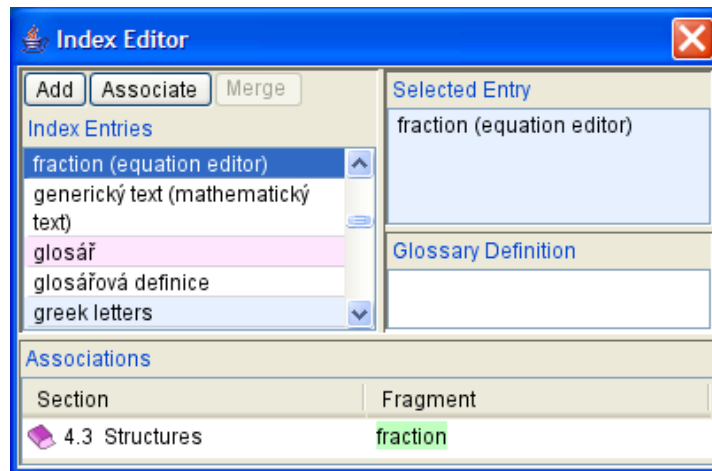
Note that the Editor supports only the master-slave scenario. That is, if you want to amend a group of localized documents, you should first amend the master version, and then update the localized versions accordingly. You cannot, for example, develop two language versions of a single document in parallel, for example write chapters 1–5 in English and 6–10 in Czech, and later on finish the missing chapters in the other language. You must also decide at the very beginning which language would play the master role.

Note that you may use the Create Localized Document command only once to generate an initial version of the localized document, and then you may change its structure considerably w.r.t the master document. However, if you want to use the automated synchronization system, note that the master and the slave documents must have identical structure, and may only differ in the paragraph texts. It is not possible, for example, to have a section with two paragraphs in the master document, which corresponds to three-paragraph section in the translated document. If you make any structural changes in the translated document, they will get lost upon next synchronization with the master document. This implies that if you delete any material from the master document, the corresponding translated material is deleted from the localized document as well. It is therefore recommended to backup your localized document before it is updated with the master.

7.2.1 Localizing the Index

If your document contains an index, you need to translate it as well. Translating index is quite similar to translating normal document content. If you open the index editor in a localized document, index entries are marked with the same colors as in the document, see obr. 7.3.

Up-to-date index entries are displayed with normal, white background. Entries needing translation have blue background, and entries needing revision have purple background. When you translate or revise the index entry, press the Commit button to confirm the translation. Note that line in the document, strict one-to-one correspondence between master and localization index entries is required.



Obř. 7.3 Index Editor Displaying Localized Index

You may add or remove index entries to the localized document, but such changes get lost when you update the slave with the master document again.

Note that when localized representation of a paragraph containing index association is added to a localized document, the associations are reconstructed in the localizable copy, and are automatically redirected to the localized index. When you translate the paragraph, you may do it in such a way that the associations are preserved. You are however free to add or remove associations, that is, index associations do not count to the requirement on structural equivalence of the master and the localized document. This also implies, that if you add new index association to a paragraph that has been localized already, that association is not reconstructed upon next localization update, and must be reconstructed manually.

Kapitola 8

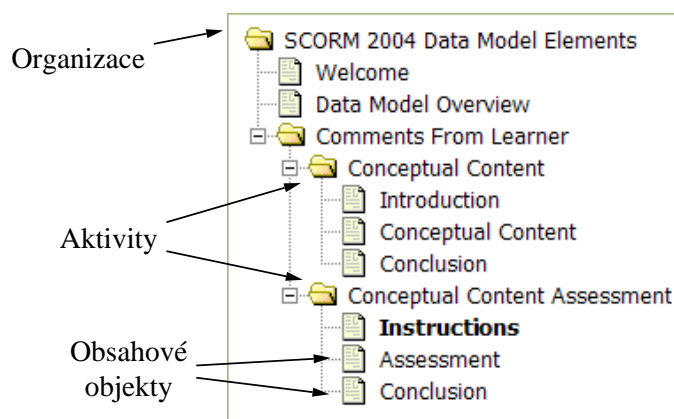
Vytváření online kurzů

Program BookEditor obsahuje nástroj pro přípravu online kurzů kompatibilních se standardem ADL™ SCORM®. Proces vytváření a instalace online kurzů je popsán v této kapitole.

8.1 O standardu SCORM

SCORM je zkratka Sharable Content Object Reference Model (referenční model pro sdílené obsahové objekty). Podle jeho autorů je SCORM "sběrka standardů a specifikací použitých z více zdrojů, poskytujících komplexní sadu e-learningových funkcí zajišťujících spolupráci, přístupnost a znovupoužití webových výukových materiálů." Jednotlivé kurzy jsou reprezentovány standardním způsobem, takže mohou být snadno importovány do různých spolupracujících systémů pro podporu výuky (Learning Management Systems, LMS).

Struktura kurzu se v terminologii SCORMu nazývá *organizace* (organization). Je to hierarchický systém *aktivit* (activities), které mohou obsahovat několik *obsahových objektů* (content objects) viz příklad na obr. 8.1.









Obr. 8.1 Struktura SCORM Kurzu

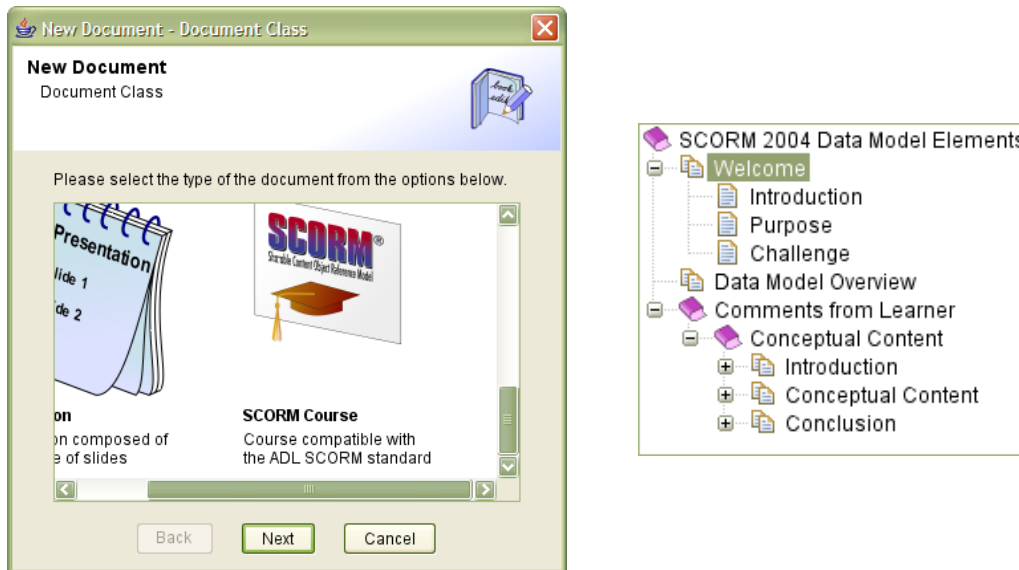
Content object is the actual media that is delivered to the learner. It is basically a web page or series of web pages. The learning management system provides facilities for navigation between content objects, or its individual pages.

The BookEditor supports creating documents conforming to the SCORM structure, and can export them directly to the SCORM format. However, note that not all SCORM features are supported by the BookEditor yet, namely

- Editing of metadata about the course.
- Changing navigation strategy from simple back-and-forth flow navigation.
- Communication with the LMS, such as in self-assessment tests.
- Alternative organization structures for single course.

8.2 Creating SCORM Course Document

To create new SCORM document, simply use the File→New... command from the pulldown menu, and select the SCORM Course document class. Then fill the document path and title as usual. Then use the global document pane to create new activities, content objects, and pages. Activities are denoted with the  icon. Activities may contain either pages () , representing single-page content objects, or multipage content objects (). The multipage objects then may contain pages. Nodes denoted with  and  icons are mere containers of other nodes, and thus have no associated documents. Page nodes denoted with  icon have associated documents, but may not contain child nodes.



Obr. 8.2 Creating and Structuring a Course Document

Kapitola 9

Language Support

This chapter discusses how the RichDoc framework supports various languages. You may easily type text in any language that is supported by your local operating system, that is, your system must have installed appropriate fonts and input facilities. However, there is more than just typing text. This chapter tells you what you need to know to prepare non-English documents.

9.1 Setting the Language

As mentioned in sekce 1.5, you select the primary language of a document when you are creating it. The language affects certain actions of the Editor, such as hyphenation, spell-checking, and text search. It also affects the language of text fragments automatically inserted to the document, such as prefixes of section titles. That is, when you change the document language from English to Czech, “Chapter 5” automatically turns to “Kapitola 5”, “Table of Contents” to “Obsah”, etc. Algorithms that perform text sorting are also affected: for instance, if rules for the Czech language are in operation, the text ‘ch’ is treated as single character during sorting, that appears between ‘h’ and ‘i’. The word ‘chata’ is thus considered after the word ‘hrad’ in alphabetical order, unlike English sorting rules.

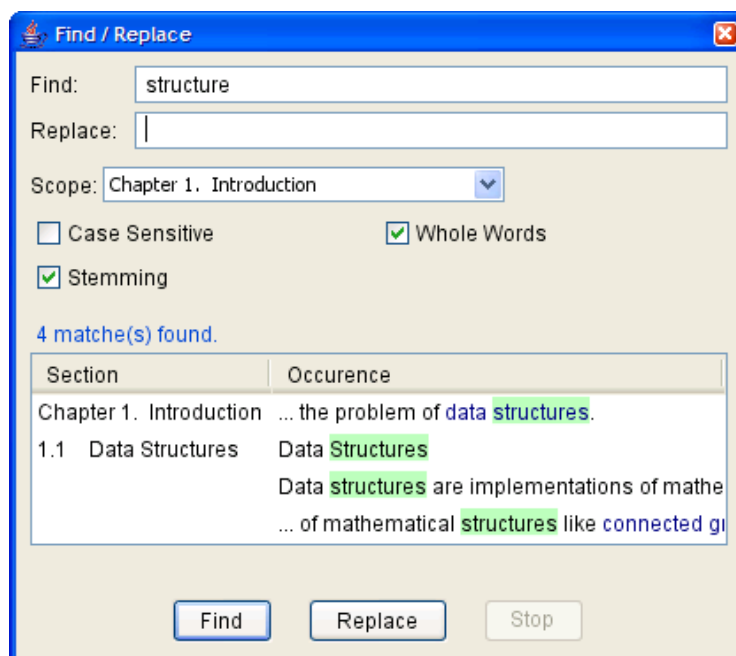
The current version of the RichDoc Framework supports only two languages: English and Czech. If you need a support for another language, you may request it from the authors, or you may try to create required resources yourself, see sekce 18.1. If the support for some language is not available, it is still good idea to set the document's language field to the correct value, just as information of potential readers or bookkeeping programs that may receive your document, or simply because the framework should not try to apply English-specific rules, such as morphology.

Kapitola 10

Finding and Replacing Text

10.1 Finding Text

Like other document-management systems, the RichDoc framework contains a built-in facility for finding and replacing text. To find or replace text, use the Find / Replace command from the popup menu, or just press Ctrl-F. A Find / Replace window appears, see obr. 10.1.



Obr. 10.1 The Find / Replace Window

The two fields at the top of the window allow you to select the text to find, and eventually the replacement text. Using the Scope field, you may select whether only the active part being edited should be searched or modified, or whether the operation should be performed on the whole document.

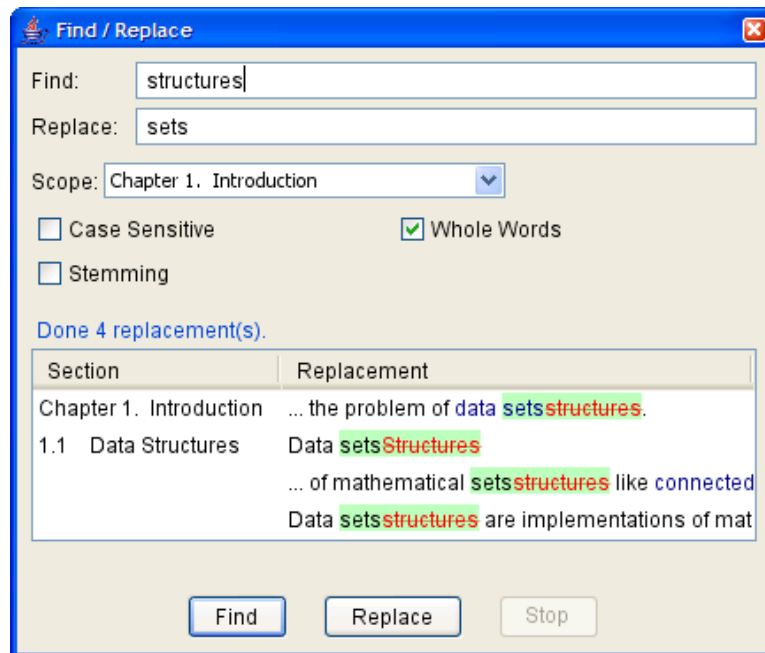
The Case Sensitive option determines whether the capitalization matters during the search. If checked, only words that exactly match to the search phrase are found, if unchecked, words that differ only in capitalization are also selected. The Whole Words option determines whether the search phrase may be matched only with whole word (separated by spaces or punctuation), or it may be matched with a part of a word. For instance, the search phrase 'text' is matched with the word 'textual' only if the Whole Words option is unchecked. The Stemming option, when checked, attempts to match words that are morphological inflections of the search phrase. For instance, if you are looking for 'cat on the mat', the phrases 'cats on the mat', 'cat on the mats', etc. are also matched, if the stemming option is enabled. Note that the stemming is only available if the Whole Words option is also enabled.

When search options are specified, you may press the Find button. After a while, the Results list is populated with the search results. The results are organized into two columns: the first column displays the name of section where the search phrase has been found, and the second column displa-

ys a fragment near the occurrence of the search phrase, with the search phrase highlighted. You may double-click a row in the list, to navigate to the corresponding place in the document.

10.2 Replacing Text

To replace the occurrences of the search phrase, specify the replacement text in the Replace field, and press the Replace button instead of the Find button. The rules for matching the text are the same as described in sekce 10.1, but additionally, all matched texts are replaced with the specified replacement. The Results list this time summarizes the replacements that have been made, see example in obr. 10.2.



Obr. 10.2 Replaced Text

As you can see in the figure, the results of the replacement operation look similar to those of the find operation. Each row shows the text that has been matched and deleted, displayed in red color and stroked-through, as well as the text that has been inserted.

Kapitola 11

Importování a exportování dokument

The RichDoc framework can import and export documents from/to several formats. It also contains a component for presenting documents on the Web using the Apache Tomcat technology. This component may be readily used as a stand-alone web application, or may be easily integrated into an existing Tomcat application.

In particular, the RichDoc framework includes quality components for importing LaTeX (see sekce 11.7) and exporting HTML (sekce 11.6). This enables to use the RichDoc as a LaTeX to HTML converter, using the RichDoc format as an intermediate format.

11.1 Rozhraní k export/import modul m

Všechny exportovací a importovací moduly mají jednotné rozhraní pro zadávání parametr konverze. Konverzi lze spustit bu přímo z hlavního okna aplikace BookEditor (z menu Soubor→Import nebo Soubor→Export), nebo z příkazové řádky.

Každý modul má jinou sadu parametr které ovliví jeho funkci. Skupiny vyplněných parametr modulu lze ukládat ve form *profil*. Pokud například exportujete určitý dokument s určitým nastavením exportovacího procesu, můžete tyto parametry uložit pod vhodným jménem, a později je znovu použít. Je-li proces konverze spuštěn z aplikace BookEditor, zobrazí se správce profilů, viz obr. 11.1.



Obr. 11.1 Okno správce profilů

At the top part of the window, there are controls for profile management. You can select active profile in the drop-down list, create new profile, save modified profile, or delete profile. Below, there are details of the selected profile. The profile consists of list of options for which you can specify custom

values. The values that are in italic (text) or are blended (check boxes) indicate default values. The red ‘A’ letter indicates that the value is computed automatically from other value(s). For instance, the Ma-in File value is associated with the Input Path value – it is the file title of the Input Path. If you change some value, the Save button becomes available, allowing you to save the modified settings. After pressing the OK button, the conversion process starts.

Alternatively, you may invoke the conversion from the command line using the command

```
richDocIo -mode mode [-profile profile] [-interactive] profile_options
```

where the *mode* may be any of values specified in tabulka 11.1. The list of options is then documented separately for each mode.

Tabulka 11.1 Export / Import Modes

Mode	Section	Mode	Section
exportHtml	sekce 11.2	importHtml	sekce 11.6
exportLatex	sekce 11.3	importLatex	sekce 11.7
exportPdf	sekce 11.4	importDocbook	sekce 11.8

11.2 Exporting HTML

This module converts RichDoc documents into the Hypertext Markup Language format. The module translates the logical structure of the RichDoc document into appropriate HTML markup, including lists, tables, hyperlinks etc. Embedded figures and formulas are automatically converted into inline images linked from generated HTML files. The document may be split into separate files at specified level. Optionally navigation may be added to the generated files. A complete list of options follows.

General

Input Path (-inputPath)

Input path to the source RichDoc document.

Output Path (-outputPath)

Output path to the desired HTML output. It may be either a directory or a ZIP file.

Character Encoding (-characterEncoding)

Desired character encoding of the generated HTML files.

Split Level (-splitLevel)

Specifies at which level sections should get split into separate HTML files. Zero value specifies the level of chapters, value 1 corresponds to the level of sections, and so on.

Condensed Code (-condensedCode)

Specifies whether the generated HTML should omit any whitespace to reduce size.

Style Sheet

Style Sheet Path (-styleSheetPath)

Path to the style sheet file. The file is put into the generated directory or ZIP file, and generated HTML files are linked to the style sheet. If not specified, default style sheet is used.

Style Sheet Local Path (-styleSheetLocalPath)

Using this option you may specify different local path to the generated stylesheet.

Navigation

Create Top Navigation (-createTopNavigation)

Add a banner to the top of each generated HTML file with links to previous, next and up sections.

Create Bottom Navigation (-createBottomNavigation)

Add a banner to the bottom of each generated HTML file with links to previous, next and up sections.

Create List Of Child Links (-createListOfChildLinks)

Add links to immediate child sections to the end of each HTML pages.

Frame Set

Create Table Of Contents Frame (-createTableOfContentsFrame)

Create a series of HTML files representing expandable Table of Contents for the document. The generated index.html file would correspond to a HTML frameset containing the Table of Contents on the left and the main document body on the right.

contentFrameName (-contentFrameName)

Name of HTML frame containing the document body.

Decoration

Bottom Line (-bottomLine)

Specify text that should be added to the end of each generated HTML page, such as authorship or copyright information.

animation

imageMagickConvert (-imageMagickConvert)

Microsoft Html Help

Create Chm File (-createChmFile)

Compile generated HTML files into Microsoft HTML Help format.

Chm Output Path (-chmOutputPath)

Hhw Title Page (-hhwTitlePage)

Hhw Executable (-hhwExecutable)

11.3 Exporting LaTeX

Sometimes you may want to export RichDoc document into the LaTeX typesetting format. For instance, you have prepared an article using the BookEditor, and you want to send it to a publisher that requires special visual style of articles, and provides LaTeX-compatible style file for that reason. You may tune BookEditor's style sheets to match the visual appearance of your document to the required style, but the easiest way is just to export the document to LaTeX, add the publisher's LaTeX style, and use LaTeX to generate the final form.

This export module is still under construction, all of the problems are not solved yet, but in general, most features of the RichDoc framework can be easily converted to LaTeX. A list of issues that deserve particular attention is listed below.

Support for non-English languages

The language of the document is used to generate the reference to the LaTeX babel package. This ensures that titles of appropriate language are used, and that correct hyphenation patterns are activated. Regarding non-English characters, two options are possible. For accented characters (such

as or ü), it is possible to convert them to latex macros, such as `\v{C}` and `\:u`. The second option is to save the document in some eight-bit encoding (such as iso-8859-1 for Western European languages, or iso-8859-2 for Central European Languages). The exported document then uses the `inputenc` package to specify the encoding.

Support for equations

Most equations should be exported to LaTeX without major problems, but some of the more exotic features may not be supported.

Support for 2D drawing

The embedded pictures are converted to the Encapsulated Postscript format, and if you have installed the GhostScript package, they may be optionally converted to the PDF format (you'll need this if you want to use PDFLaTeX to process your LaTeX document.) There is also a problem with non-English characters, because PostScript format does not have any uniform support for handling non-English characters. In this case, the export module converts words containing non-English characters to curves. This approach usually yields acceptable results.

Support for Tables

Converting complex tables to LaTeX may cause major problems, as the LaTeX table layout algorithm has quite limited capabilities. It cannot, for example, automatically determine appropriate width of columns containing long paragraphs. In fact, if you want to have a word-wrapped paragraph in a table, you must set the width of the table column to some fixed value.

Support for Index and Bibliography

Index and bibliography is automatically converted to LaTeX conventions. Moreover, the finishing phase, if enabled, automatically calls `bibtex` and `mkindex` programs to make index and bibliography correct. LaTeX-style glossary is not yet supported.

A complete list of options follows.

General

Input Path (`-inputPath`)

Input path to the source RichDoc document.

Output Path (`-outputPath`)

Output path to the desired LaTeX output. It may be either a directory or a ZIP file.

Main File (`-mainFile`)

Name of the file corresponding to the LaTeX document root, i.e. file to be processed with LaTeX.

Character Encoding (`-characterEncoding`)

Specifies character encoding to be used for non-ASCII characters. If "LaTeX" encoding is selected, non-English characters are escaped using common TeX/LaTeX convention, e.g. ü (u with umlaut) is escaped as `\:u`. If other encoding is selected, characters are encoded into single bytes, and an appropriate command is added to the document preamble.

Local Path (`-localPath`)

Specifies a path offset from the root file to other files. This path is used to generate inclusion commands, such as `\include` or `\includegraphics`.

Process Embedded Images (`-processDrawings`)

Specifies whether Encapsulated Postscript files should be generated for embedded 2D pictures. This option may be useful for temporarily disabling picture generation, if a large document needed to be converted several times and picture generation slows down the conversion too much.

Process Inline Bitmaps (-processInlineBitmaps)

Specifies whether Encapsulated Postscript files should be generated for embedded inline bitmaps, see also Process Embedded Images option.

Generate PDF For Embedded Images (-generatePdfForEmbedded)

Generates embedded images in the PDF format besides the Encapsulated Postscript format. This is needed if you want to process the generated LaTeX document with PDFLaTeX to generate a PDF file.

Options**Font Size (-fontSize)**

Specifies the font size of the target document.

Sloppy (-sloppy)

Specifies whether LaTeX `\sloppy` mode should be turned on. This mode ensures that LaTeX breaks paragraphs even if the breaking creates very large spaces between words. When disabled, problematic word is not put on the next line, but rather exceeds the printable area.

Max Width (-maxWidth)

Specifies the maximum width of tables and figures, in pts (1pt = 0.35mm or 1/72"). If a figure or table exceeds the maximum width, it is automatically scaled down to fit into the width.

Max Height (-maxHeight)

Specifies the maximum height of tables and figures, in pts (1pt = 0.35mm or 1/72"). If a figure or table exceeds the maximum height, it is automatically scaled down to fit into the height.

Finishing**Finishing Mode (-finishingMode)**

Specifies whether the generated LaTeX document should be processed with LaTeX to generate a DVI file, with LaTeX + DVIPS to generate a PostScript file, or PDFLaTeX to generate a PDF file.

Finish Dir (-finishDir)

Specifies the directory where the LaTeX processor is invoked. Changing the directory may be useful if you want to use main file different from the automatically generated main file.

Finish Main File (-finishMainFile)

Specifies the directory for which the LaTeX processor is invoked.

11.4 Exporting PDF

This module can be used to convert a RichDoc document into a PDF format. The result is similar as if the document is printed to a regular printer. Optionally, hyperlinks in the document are converted to PDF hyperlinks, and/or an interactive table of contents (called bookmarks in PDF terminology) is added to the PDF document.

General**Input Path (-inputPath)**

Input path to the source RichDoc document.

Output Path (-outputPath)

Output path to the desired PDF output.

Options

orientation (-orientation)

paperSize (-paperSize)

Hyperlinks (-hyperlinks)

Whether RichDoc hyperlinks should be converted to PDF hyperlinks.

Bookmarks (-bookmarks)

Whether PDF bookmarks, i.e. interactive table of contents, should be added to the PDF document.

Incremental (-incremental)

11.5 Exporting SCORM

This module converts RichDoc document into a file conforming to the ADL SCORM standard.

11.6 Importing HTML

This module converts HTML documents into a RichDoc document.

General

Input Path (-inputPath)

Specifies the path to the HTML files to be imported. The path may be either a directory containing HTML files, a ZIP file containing HTML files, or output from the KSMSA Web Crawler.

Output Path (-outputPath)

Specifies the path to the RichDoc document to be generated.

Language (-language)

Specifies the primary language of the generated RichDoc document.

Character Encoding (-characterEncoding)

Specifies the character encoding of the input HTML documents, if they contain encoded non-ASCII characters. Note that known HTML entities corresponding to non-ASCII characters (such as ü) are automatically converted to single UNICODE characters. If encoding is specified in the Web Crawler database (i.e. HTML document was downloaded from the web and the web server returned encoding for it in the HTTP response header), that value is used instead. Note that encoding value from the http-equiv header tag is not used.

Document Class (-documentClass)

Specifies the desired document class of the target document.

File Filter

Include File (-includeFile)

Specifies a list of HTML files that should be imported. You may use wildcards, such as *.html for all files with html extension in the main directory, or **/*.html for all HTML files in all directories. If this list is empty, all files are included.

Exclude File (-excludeFile)

Specifies list of files that should not be imported. You may use wildcards.

Content Filter

Content Start (-contentStart)

Specifies a regular expression defining a position in a file where the conversion should start, such as `<body>`. This is useful to omit e.g. navigation code or banners. If not specified, or the start sequence is not found in the file, conversion starts from the beginning of the file. Otherwise, the conversion starts from the first occurrence of the start sequence.

Include Content Start (-includeContentStart)

Specifies whether the start sequence should be converted.

Content End (-contentEnd)

Specifies a regular expression defining a position in a file where the conversion should end, such as `</body>`. If not specified, or the end sequence is not found in the file, conversion continues to the end of the file. Otherwise, the conversion ends at the last occurrence of the end sequence.

Include Content End (-includeContentEnd)

Specifies whether the end sequence should be converted.

Exclude Fragment (-excludeFragment)

Specifies a regular expression of fragments that should be excluded from the conversion process.

Replace Fragment (-replaceFragment)

Specifies a regular expression-replacement pairs that should be used to pre-process the input HTML file. All occurrences of the regular expressions are replaced with the corresponding replacement.

Ignore Tags (-ignoreTags)

List of tag names (without `<` and `>` delimiters) that should be ignored. Note that only the tag delimiters are ignored, the content of the tag is processed normally. If you want to ignore entire tag, use Exclude Fragment option with regular expression `<tag[^>]>.*</tag>`.

Output Filtered (-outputFiltered)

Specifies a path to a ZIP file that will contain HTML files that were actually used for conversion, when filtering and replacements were applied. Since any problems found during conversion are reported w.r.t. the filtered content, the filtered files may be useful to

Content Modification**Number Sections (-numberSections)**

Whether the sections generated from the `<h*>` tags should be numbered by the RichDoc framework.

Remove Original Section Numbers (-removeOriginalSectionNumbers)

Whether original section numbers from titles should be removed. This option should be checked if you also checked the Number Sections option.

Detect Unnumbered Sections (-detectUnnumberedSections)

This option marks sections that didn't contain number in original HTML source as unnumbered. Otherwise, they are automatically numbered unless you disabled the Number Sections option.

Misc**Print Font Size (-printFontSize)**

Specifies the font size used when the generated RichDoc document is printed. If not specified the default font size is used.

11.7 Importing LaTeX

Documentation not yet available.

General

Input Path (-inputPath)

Specifies the path to the main LaTeX file to import.

Output Path (-outputPath)

Specifies the path to the RichDoc document to be generated.

Language (-language)

Specifies the primary language of the generated RichDoc document.

Character Encoding (-characterEncoding)

Specifies the character encoding of the input LaTeX document, if it contains encoded non-ASCII characters. Note that LaTeX-escaped non-ASCII characters (such as $\backslash\{u\}$) are automatically converted to single UNICODE characters.

11.8 Importing DocBook

This module imports documents obeying the DocBook [1] XML markup, see <http://www.docbook.org>.

General

Input Path (-inputPath)

Specifies the path to the XML files with DocBook markup to be imported. The path may be either a directory containing XML files, a ZIP file containing HTML files, or output from the KSMSA Web Crawler.

Main File (-mainFile)

The main XML file corresponding to the root of the DocBook document.

Output Path (-outputPath)

Specifies the path to the RichDoc document to be generated.

Language (-language)

Specifies the primary language of the generated RichDoc document.

Character Encoding (-characterEncoding)

Specifies the character encoding of the input HTML documents, if they contain encoded non-ASCII characters.

Document Class (-documentClass)

Specifies the document class of the generated RichDoc document.

Create TOC (-createToc)

Whether Table of Contents should be added to the generated RichDoc document.

Create Short TOC (-createshortToc)

Whether short Table of Contents (chapters only) should be added to the generated RichDoc document.

File Filter**Include File (-includeFile)**

Specifies a list of files or wildcards to be imported.

Exclude File (-excludeFile)

Specifies a list of files or wildcards to be excluded from the conversion process.

Misc**Print Font Size (-printFontSize)**

Specifies the font size used when the generated RichDoc document is printed. If not specified the default font size is used.

11.9 Deploying Documents

Documentation not yet available.

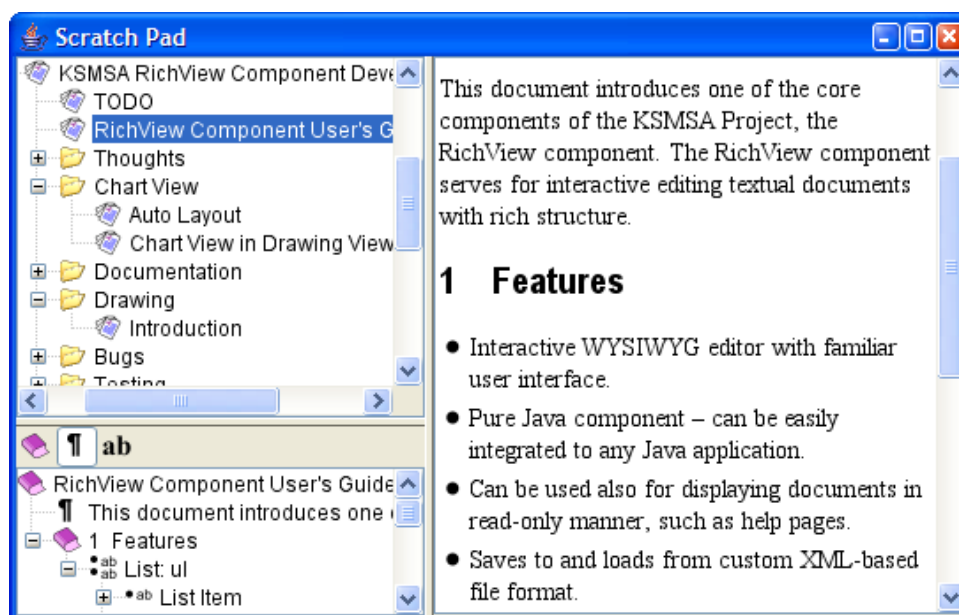
Kapitola 12

Printing

Kapitola 13

The ScratchPad Application

So far, we discussed the primary application of the RichDoc framework, the BookEditor. The RichDoc framework also contains the ScratchPad application, intended for writing and managing personal notes. The user interface of the ScratchPad is quite similar to that of BookEditor, see obr. 13.1.



Obr. 13.1 The ScratchPad Application

As you can see in the figure, the user interface is the same, except the upper-left pane, which does not display the structure of single document, but the structure of your personal notes.

13.1 Managing the System of Notes

The Notes Pane shows a hierarchical list of your notes. The items denoted by 📁 are groups, that contain other groups, or notes. Notes are denoted by the 📄 icon. The selected note may be denoted by the 📄 icon to indicate that it has been modified and needs saving.

If a group is denoted by the 📄 icon, it means that the group itself has an associated note. A group may also be denoted by the 📁 icon, which indicates that when it is selected, its children are collected to displayed in the editor pane as a read-only document.

Kapitola 14

Troubleshooting

The RichDoc framework is an experimental project that is under rapid development. This means that the framework may contain bugs, many features are not finished, and some features and file formats may be changed without notice. If you still decide to use it on a production level, you should be very careful, save and backup your files regularly, and be prepared to face frequent problems. Namely, the framework has the following limitations:

- There is no undo/redo support. What you do to the document cannot be easily reversed. You should thus save your document frequently, so that you can revert to previously saved version if your document gets damaged, either due to your mistake, or due to the failure of the framework. If your document gets damaged during saving, you may attempt to recover it from emergency backup file, as described in sekce 14.2.

14.1 Platform-specific problems

Although the RichDoc framework is implemented using technology supporting all major operating systems, there are some known problems with certain platforms. This section discusses some of them.

- On certain systems, it is difficult to set up input facilities to support national (non-ASCII) characters. For this purpose, the BookEditor application contains a user-level feature that maps keys on a keyboard to other, non-English letters. With this feature, you may type text in non-English language as if the underlying system supported it. The character map is currently provided only for the Czech language. You can switch between English and Czech keyboard by clicking the language button in the bottom-right corner of the main frame.
- The framework relies on the underlying window system to send notification when the main application window is deactivated. For instance, the main toolbar should be hidden upon deactivation. However, some systems fail to send such notification. For instance, if you use the cygwin X-server under MS Windows to display a RichDoc application running on UNIX system, deactivating the RichDoc application displayed in X-windows frame, by activating other window that itself is not an X-windows, does not cause the deactivated window to be notified. This causes the main toolbar window to remain visible, obscuring the newly activated window. This problem can be fixed only by configuring the RichDoc framework to display the main toolbar as a *lightweight* window. In lightweight mode, the toolbar is not a real window, but just a rectangular area painted into the host window, namely the RichDoc main frame. This solves the problem, but prevents the main toolbar to be dragged off the main frame of the RichDoc application.
- Some systems, namely cygwin X-server running under MS Windows, may not support drag & drop operations. There is again a possibility of enabling a *lightweight* drag & drop, which handles mouse events and simulates drag & drop operation on application level. This effectively enables the drag & drop, albeit with limited functionality.

14.2 File Backup and Recovery

Each time you save a document, the book editor creates an incremental backup of the file before it is rewritten with the new version. The word *incremental* refers to the fact that only files

that are going to be modified are backed up. This keeps the size of backup files small. The backup files are ZIP files containing original content of modified ZIP files. The emergency file has the path *\$TEMP/ksmsa/richDoc/backup/\$FILE-\$ID/\$DATE.zip*, where *\$TEMP* is the path to your system's temporary directory (usually *c:\Documents and Settings\%USERNAME%\Local Settings\Temp* under Windows), *\$FILE* is the short name of the file being saved, *\$ID* is the global identifier of the file main section, and *\$DATE* is the date of backup.

Note that this feature is intended for emergency purposes only, you should not rely on it as a kind of version management system. A better, more user friendly version management system is under construction.

Kapitola 15

Acknowledgments

The RichDoc framework is an open-source project, which draws resources and libraries from other open-source systems. In this chapter, we would like to thank to all contributors for their willing to share their resources. tabulka 15.1 summarizes projects and resources that have been adopted.

Tabulka 15.1 Resources used in the RichDoc Framework

Consortium	Project	Comment & Copyright
Apache Software Foundation	Apache XML Graphics	PDF generation library. Copyright (C) 1999-2003 The Apache Software Foundation. This product includes software developed by the Apache Software Foundation (http://www.apache.org/).
	Apache Ant	Packaging and compressing utilities (TAR, bzip2).
TeX Users Group	TeX	Math fonts
		English hyphenation patterns (c) Frank Liang.
		Czech hyphenation patterns (c) Pavel Ševe ek, Lingea s.r.o.
		Babel Package, Copyright 1993–2005 Johannes L. Braams & Contributors, under the LaTeX Project Public License.
ispell contributors	ispell	English spell-checking database (c) by Geoff Kuenning and other unpaid contributors.
		Czech spell-checking database (c) 2001 by Petr Kolá . Contributors to the Czech dictionary: Tomáš ermák, Petr Prenghy, Hanuš Adler, Petr Kolá .

ást II

Expert's Guide

Kapitola 16

Introduction to the Data Model

Kapitola 17

The RichDoc Print Format

The RichDoc contains a facility for printing documents into intermediate format, that can be used later for print preview or sending to a real printer. The intermediate files are useful to avoid repetitive print layouts, which may be very lengthy for big documents.

The RichDoc Print format has similar role like PostScript and PDF formats, and also has similar architecture. There are several reasons why we decided to create a new format and not to reuse existing standard formats. The main reason is that there is no cross-platform, patent free, Java™ compatible library for writing and reading PostScript, PDF or similar standard format. Implementing such library would be difficult, as the formats are very complex, and many of their features could not be even utilized by our framework. Another reason is that these formats do not have good support for Unicode, while Java™ Printing APIs fully support Unicode.

Moreover, our format has one more extra feature: it supports *incremental printing*, which allows modification of existing print file by reprinting only those pages that have been modified since last printing. This feature is useful for keeping print files of large documents up-to-date, if the documents are often modified.

The RichDoc Print format is merely a serialization of commands issued through the interface of the `java.awt.Graphics2D` class. That is, the “printer” component implements the `Graphics2D` interface by serializing the commands and storing them to a binary file. The “playback” component reads the serialized commands, and sends them to supplied object compatible with `Graphics2D`, which may be either real printer or a print-preview component.

17.1 The Overall Structure of a Print File

The print file is actually a ZIP file, consisting of page definition files, embedded bitmaps, document source title, and an index to support the incremental printing function. Each page definition file obeys a format described in sekce 17.2.

17.2 The Page Description Format

This section defines the format of a single file within the overall print file, which is a definition of objects on a single page. The page-definition format is binary, i.e. it is a stream of bytes. Therefore, we need to define data types that are used to define object properties, and how they are serialized into streams of bytes. The data types are defined in sekce 17.2.1.

tabulka 17.1 defines the top-level structure of the page definition file. First, the description of the printing media is written, see sekce 17.2.2. Then follows a list of printing instructions, as they were recorded by the printer component. Each instruction starts with a code byte, which defines the type of the instruction, see sekce 17.2.3. The sequence of instructions is terminated by a `END_OF_FILE` value is encountered.

17.2.1 Data Types

The data types are listed in tabulka 17.2. All integer data types are signed using the common, i.e. “two’s complement”, encoding of negative numbers (e.g. unsigned `0xFF` represents `-1` byte value, `0xFE` `-2` value, etc.) All integers are stored in *big endian*, that is, more significant values are stored first.

Tabulka 17.1 Overall Page Definition Structure

Field	Description
pageFormat	description of the printing media, see tabulka 17.8
byte	#1 instruction code
instruction	#1 instruction data, see tabulka 17.9
byte	#2 instruction code, etc., until END_OF_FILE byte encountered

Real numbers are encoded by first converting them to integers using `java.lang.Double.doubleToLongBits()` or `java.lang.Float.floatToIntBits()`, and then encoding them in big endian as integers. This encoding corresponds to the IEEE 754 standard for encoding numbers.

The string value is encoded the same way as if saved using `java.io.RandomAccessFile.writeUTF()`. That is, the length of the string is first encoded as a 16-bit integer, and then the string characters are appended, each encoded using the UTF-8 encoding.

Tabulka 17.2 Data Types

Name	Description
byte	8-bit signed integer
double	64-bit real number
float	32-bit real number
short	16-bit signed integer
int	32-bit signed integer
long	64-bit signed integer
boolean	8-bit boolean
string	UTF-8 encoded string, with leading short for the string length
shape	serialization of <code>java.awt.Shape</code> , see tabulka 17.3
stroke	serialization of <code>java.awt.BasicStroke</code> , see tabulka 17.4
transform	serialization of <code>java.awt.geom.AffineTransform</code> , see tabulka 17.5

Besides primitive data types, there are also more complex types needed to encode some parameters of the `Graphics2D` interface. The serialization of `java.awt.Shape` is described in tabulka 17.3.

Tabulka 17.4 Serialization of `java.awt.BasicStroke`

Field	Description
float	line-width
byte	end-cap: 0 – butt, 1 – round, 2 – square
byte	join-type: 0 – miter, 1 – round, 2 – bevel
float	miter-limit
short	dash length
float	dash #1
float	dash #2 etc.
float	dash phase (only if dash length > 0)

Tabulka 17.3 Serialization of `java.awt.Shape`

Field	Description	
byte	winding rule, 0 – even-odd, 1 – non-zero	
byte	type of #1 segment, 0 – move-to, 1 – line-to, 2 – quad-to, 3 – cubic-to, 4 – close	
data for segment #1		
byte	type of segment #2, etc, until -1 is encountered	
move-to, line-to	float	x endpoint coordinate
	float	y endpoint coordinate
quad-to	float	x control point coordinate
	float	y control point coordinate
	float	x endpoint coordinate
	float	y endpoint coordinate
cubic-to	float	x control point #1 coordinate
	float	y control point #1 coordinate
	float	x control point #2 coordinate
	float	y control point #2 coordinate
	float	x endpoint coordinate
	float	y endpoint coordinate

Tabulka 17.5 Serialization of `java.awt.geom.AffineTransform`

Field	Description
double	the scale-x value of the transform matrix
double	the shear-y value of the transform matrix
double	the shear-x value of the transform matrix
double	the scale-y value of the transform matrix
double	translate-x value of the transform matrix
double	translate-y value of the transform matrix

Tabulka 17.6 Serialization of `java.awt.Color`

Field	Description
byte	the alpha value (0 = transparent, 255 = opaque)
byte	the red value
byte	the green value
byte	the blue value

17.2.2 Printing Media Definition

The printing media definition section of the file defines the size and orientation of the paper to which the material is printed, see tabulka 17.8. First, the orientation of the paper is written, then its to-

Tabulka 17.7 Serialization of `java.awt.Font`

Field	Description
string	font name
byte	font style (0 – plain, 1 – bold, 2 – italic, 3 – bold italic)
float	font size

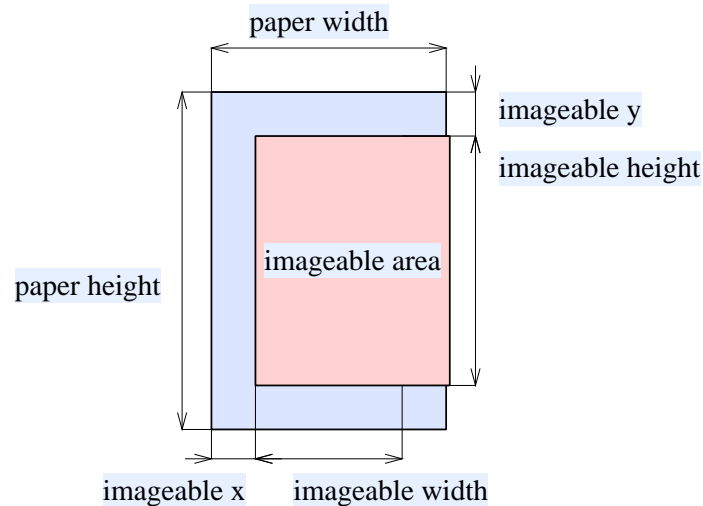
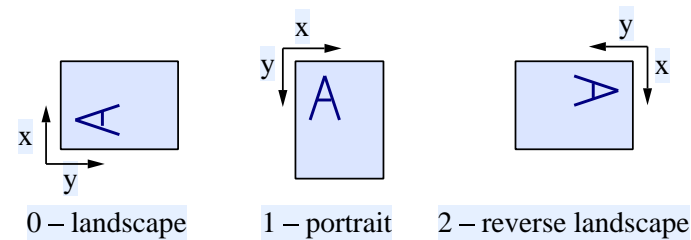
tal size, imageable size (total size without margins), and imageable area offset from the total area. See obr. 17.1 for the meaning of fields in tabulka 17.8. Note that the fields describing the paper size refer to the portrait orientation of the paper. For instance, the A4 paper will always have the width of 210mm and the height of 297mm, regardless the paper orientation.

Tabulka 17.8 `pageFormat` Field Definition, see obr. 17.1

Field	Description
byte	paper orientation: 0 – landscape (Windows), 1 – portrait, 2 – reverse (Macintosh) landscape
double	paper width
double	paper height
double	imageable x
double	imageable y
double	imageable width
double	imageable height

17.2.3 Printing Instructions

We call a single call to the `Graphics2D` interface a *printing instruction*. To encode the printing instruction, we must encode its type, and all its parameters. All supported printing instructions and their parameters are summarized in tabulka 17.9. Since the parameters closely correspond to the parameters of `Graphics2D` methods, we comment them very sparsely. See documentation of `Graphics2D` for more information.



Obr. 17.1 Meaning of `pageFormat` Fields

17.3 Notes on Printing to the RichDoc Print Format

In this section, we would like to mention some limitations and other technical notes regarding using the RichDoc Print interface from a Java™ application.

First, note that some features of the `java.awt.Graphics2D` may not be implemented. For example, painting using general `Paint` component is not supported. Also note that like Java Serialization format, this format is fragile, and is subject to change without notice. You should use it only for intermediate storage of print files, not for long-term storage. No support is provided for maintaining compatibility of various versions of this format. If you detect that a print file has different format version than you expect, you should consider the file invalid, and should not attempt to parse it.

Be aware that recorded printing commands may be played back in a different context that they were captured. It is therefore illegal to use any commands that set the global state of the `java.awt.Graphics2D` class, such as by calling `setTransform()` or `setClip()`. You should use their relative equivalents, i.e. `transform()` and `clip()`. It is, however, legal to save the state of the class, e.g. by calling `getTransform`, and restore it later, e.g. by `setTransform()`. The printing component automatically converts these commands to `SAVE_TRANSFORM` and `RESTORE_TRANSFORM` instructions, respectively.

Tabulka 17.9 instruction Fields

Instruction Code	Field	Description
0 = DRAW_STRING_INT	draw string on integer coordinates	
	string	the string
	int	the x coordinate
	int	the y coordinate
1 = DRAW_STRING_FLOAT	draw string on real coordinates	
	string	the string
	float	the x coordinate
	float	the y coordinate
2 = FILL_SHAPE	fills shape	
	shape	the shape to fill
3 = DRAW_SHAPE	draws shape	
	shape	the shape to draw
4 = SET_STROKE	sets the stroke	
	stroke	the stroke to set
5 = TRANSFORM	appends transform to current transform	
	transform	transform to append
6 = SAVE_TRANSFORM	saves current transform under given ID	
	short	transform ID
7 = RESTORE_TRANSFORM	restore previously saved transform	
	short	transform ID
8 = CLIP	reduces current clip area by given shape	
	shape	the shape to clip to
9 = SAVE_CLIP	saves current clip under given ID	
	short	transform ID
10 = RESTORE_CLIP	restore previously saved clip	
	short	transform ID
11 = RESET_CLIP	resets the clip to the state before playback has started	
12 = SET_COLOR	sets current color	
	color	the color to set
13 = SET_FONT	sets the current font	
	font	the font to set
14 = SET_FONT_VARIANT	sets the variant of the current font (to save space, if the font name is the same)	
	byte	font style
	float	font size
15 = DRAW_TRANSFORMED_IMAGE	draws image with transformation	
	string	the image file name
	transform	the transform to apply before drawing
16 = DRAW_IMAGE	draws image	
	string	the image file name
17 = SAVE_SHAPE	fills given shape, and saves it under given name	
	string	shape name
	double	shape x-location
	double	shape y-location
	boolean	x-mirrored
18 = USE_SHAPE	fills previously saved shape	
	string	shape name
	double	shape x-location
	double	shape y-location
	boolean	x-mirrored
19 = SHAPE_SCALE	writes the scale to be used for SAVE_SHAPE and USE_SHAPE	
	double	scale
20 = END_OF_FILE	end of file mark	

Kapitola 18

Contributing to the RichDoc Framework

The RichDoc framework is a public-domain project, which is provided for free to its users in the hope it will be useful. The users are encouraged to contribute to the project in any way, by reporting bugs, submitting suggestions, adding code or adding localized resources.

18.1 Contributing Localized Resources

So far, the framework fully supports the English language, and partially supports the Czech language (unfortunately the only two languages the author can speak).

18.1.1 Localizing User Interface

The RichDoc framework is delivered by means of a series of JAR (Java Archive) files, that contain the executable code, as well as localizable resources for the user interface of the framework.

The most important part, that must be localized at a minimum to reasonably support a language, is in the package `org.kmsma.richView.model`. It contains names of text fragments automatically inserted to documents, such as “Table of Contents”, “Chapter”, “Section” etc.

18.1.2 Creating Language Packs

A *Language Pack* is a bundle of resources that is external to the executable JAR files of the framework. It contains the following resources:

Hyphenation Patterns

Hyphenation patterns tell the RichDoc framework how to correctly hyphenate words. The format the RichDoc framework is using is similar to hyphenation files of the TeX system, and thus can be simply imported from TeX, as hyphenation files within the TeX system are usually in the Public Domain.

Spell-checking Database

Spell-checking database allows the RichDoc framework to report spelling errors. The databases are adopted from the *ispell* project.

Stemming Rules Database

Stemming rules define how given language inflects words. It is used by the framework during indexing and search, to improve the recall of the search system. The search system can then find words even if they appear in the document in their inflected form. The database also contains the *stop list*, i.e. list of words that are not significant for the search, which includes articles, conjunctions, auxiliary verbs, etc. Database may also contain list of exceptions, i.e. list of words and their inflections that do not obey the rules.

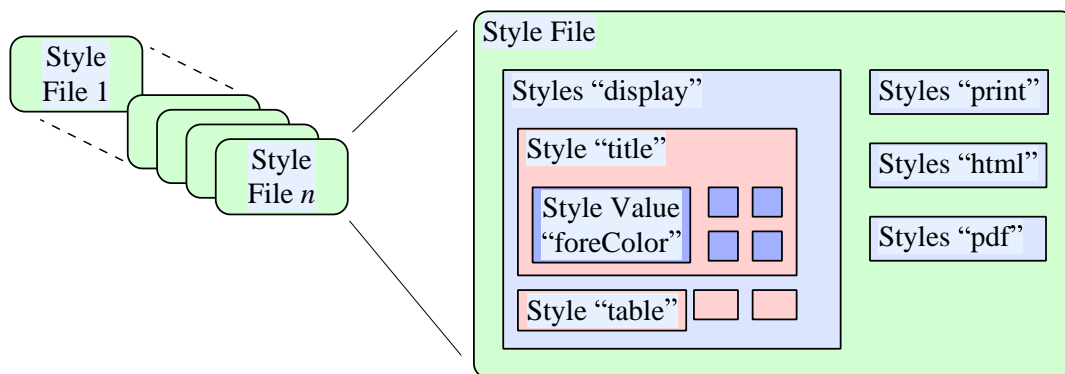
Kapitola 19

Styles

This chapter describes in more detail the style languages that are used to customize the RichDoc framework.

19.1 Visual Styles

Visual style language defines the rules that affect the process of visualization of a RichDoc document. It is quite similar to HTML Cascading Style Sheet system. The visual styles are organized into a cascade of style files. The anatomy of a style file is shown in obr. 19.1.



Obr. 19.1 Anatomy of Style File

Each style file contains visual rules, organized into structures of several levels:

Styles Level

The Styles structure groups rules applicable for different context of rendering, such as display, printing, exporting to PDF, etc. You may also create intermediate Styles structures containing general rules and inherit these from more specific Styles structures.

Style Level

The Style structure groups rules to be applicable to one element of the document, such as paragraph, table, word etc. Style structures have names that are used to match the structures with the object, see sekce 19.1.1. Like with styles structures, Style structures may inherit data from other Style structures.

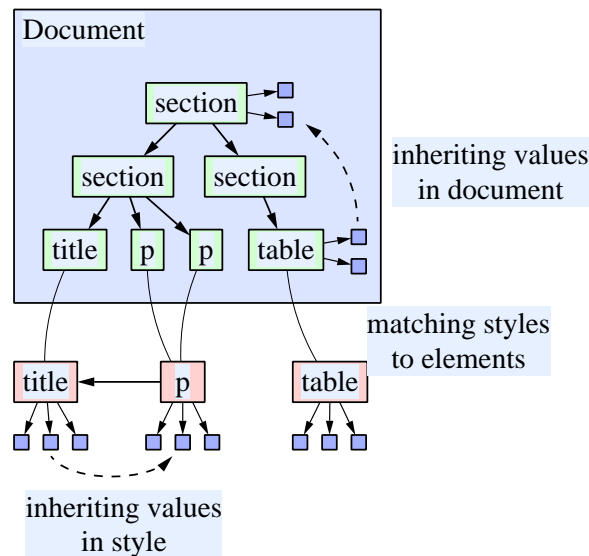
Style Value Level

The Style Value represents single visual property to be applied to an object, such as foreground color, font size etc.

19.1.1 Matching and Inheriting Style Structures

The ultimate goal of the style mechanism is to assign to each document element, such as paragraph or table, a set of style values, such as foreground color or font size. When this assignment is done, we can use the style values to render the document element. Although we may do the assignment expli-

citly for each element, like in traditional word processors, we do it more systematically using external matching mechanism, see obr. 19.2.



Obr. 19.2 Matching Elements to Styles

First, for each document element, we find the best matching Style structure. Since Style structures are actually containers of style values, this assignment effectively assigns the document element a group of style values. If we need a style value for particular element, we first ask its associated Style structure. If the value is not found, we attempt to *inherit* it from elsewhere.

If the Style structure has a parent Style structure (such as the “title” style in our example is derived from the “p” style), we first try to inherit the value from here. This kind of inheritance we call *inheritance in style*. If this fails, we attempt to inherit the value from the containing element (such as the “table” element in our example inherits from its containing “section” elements). This process we call *inheritance in document*.

Note that not all style values may be inherited in document. For instance, the margins of elements are not inherited, because that would add to the margin for each new level of containment in the document. The colors, on the other hand, are inherited in document: if we say that a section has specific foreground color, all its children inherit that color unless they override it.

Finally, we need to explain how we match Style structures to document elements. Each document has a tag name, and optional class name. Tag names of various types of elements are described in Section 17.2. They roughly correspond to HTML tag names. From the tag name and class name, we construct *element name*, which is either *tagName* or *tagName#className*. The *element path* is dot-separated list of element names of all parent elements of given element from the document root towards the element, including the final element. For instance, the element path of the table in obr. 19.2 is *section.section.table*. A *partial element path* is element path with some parts omitted. We may omit any number of dot-separated items from the beginning of the path, and/or any number of class qualifiers. For instance, the partial element paths of the path *section#appendix.p#note* are: *p*, *p#note*, *section.p*, *section.p#note*, and *section#appendix.p*. The element path or any of element partial paths are said to *match* the element. We say that p_1 is *stronger match* than p_2 if (a) p_1 is longer than p_2 in terms of the number of the dot-separated elements, or (b), if the paths have the same length, but p_1 has class qualifier sooner than p_2 . The partial paths of our example path are listed in order of their strength, from the weakest one to the strongest one.

After these complicated definitions, we come to a simple conclusion: we match document element with that style whose name has best match with the element's path. For instance, we have style

s_1 with name `section#appendix.p` and s_2 with name `section.p#note`. They both match our example element, but we match it with s_1 , because it has stronger match. That is, we ignore the `note` class of the paragraph in favor of obeying the `appendix` class of the containing section.

19.2 Document Styles

Document style language defines various document classes (for books, articles, etc), and for each class defines rules how document material is named and numbered.

19.3 Bibliography Styles

Bibliography style language defines types of bibliographical references (see tabulka 6.1), and for each reference, how the linear textual representation, that is displayed in the Bibliography section of a document, is constructed from the database fields.

Bibliography style is defined in a XML file `style.xml`. The file has structure shown in obr. 19.3. The root element is `styles`, having `style` element for each type of publication. The `type` attribute of the `style` element corresponds to one of the publication type from tabulka 6.1. Its value is the English name of the publication type in Java convention, e.g. “technicalReport”.

```
<styles>
  <style type="type">markup <$style>
  :
  <style type="type">markup <$style>
</styles>
```

Obr. 19.3 Structure of a Bibliography Style File

The markup contains the actual XML markup that generates the citation text. It may contain ordinary paragraph markup (see Section 17.2), plus the following special elements:

<dbField [b="text"] [a="text"] [from="record"] />

element that inserts one field from the record being rendered. *dbField* is the name of the field to be inserted, i.e. one of fields listed in tabulka 6.2. Again, the field name is formed from the field's English name in Java style, e.g. `<howPublished />`. The `b` attribute may be used to insert text (e.g. punctuation) before the field text, if the field text is not empty. Likewise, the value of the `a` attribute is conditionally inserted after the field text. The `from` attribute may be used for qualification of record associated with the record being rendered. For instance, if we render In Proceedings record, the `title` element refers to the in proceedings title. If we want to refer to the title of the containing proceedings, we should use `<title from="pd" />`.

If the field contains multiple values (such as the author field), all values are inserted with appropriate punctuation used to separate the values.

<name type="type" />

Inserts the localized name of the publication type. For instance, `<name type="phdThesis" />` inserts the phrase “PhD Thesis” in an appropriate language.

<s value="key" />

Inserts the localized string of any kind. The *key* identifies the string to be added. For instance, `<s value="and" />` inserts the conjunction “and” in an appropriate language.

For instance, the `inProceedings` type of publication is defined by markup “`<author />. <title />. <s value="inProceedings" /> <i><title from="pd" /></i>, <address from="pd" /> <month from="pd" /> <year from="pd" />.`”. When interpreted, it may be translated into text “Michal Šev enko and He man Mann. Intelligent user-support system for modeling and simulation. In *Proceedings of 2002 IEEE CCA/CACSD Conference*, Glasgow 2001.”.

Index

- aktivita (SCORM) 41
- animace 26
- automatické rozvržení (grafický editor) 23
- balík jazykové podpory 68
- Bézierova křivka 20
- bibliografické odkazy 32
- BibTeX 36
- bitmapa 22
- BookEditor 6
- content object (SCORM) 41
- čára 20
- číslo (matematický text) 13
- číslovaná rovnice 18
- článek (tvoří dokumentu) 7
- databáze
 - bibliografických odkaz 32, 36
- DocBook import 53
- dokument
 - jazyk 7
 - tvoří dokumentu 7
- export
 - HTML 47
 - LaTeX 48
 - PDF 50
 - SCORM 51
- externí hypertextový odkaz 11
- formátování odstavce 10
- generický text (matematický text) 13
- glosář 30
- glosářová definice 30
- hledání textu 44
- HTML
 - export 47
 - import 51
- hypertextový odkaz 11
 - externí 11
- import
 - DocBook 53
 - HTML 51
 - LaTeX 53
- index (editor rovnic) 16
- inline rovnice 13
- integrál (editor rovnic) 16
- kniha (tvoří dokumentu) 7
- kniha s částmi (tvoří dokumentu) 7
- konektor 22
- konstanta (matematický text) 13
- kopírování 9
- kreslicí nástroj 20
- klíčový odkaz 11
- KSMSA 5
- kvadratická křivka 20
- LaTeX
 - export 48
 - import 53
- Learning management system 41
- limity (editor rovnic) 16
- lineární kóta 23
- lokalizace 38

- LVM 38
- manipula ní bod (kreslení) 21
- matematický text 13
- matice (matematický text) 13
- m ení textu (kreslení) 24
- náhrada textu 45
- neuspo ádaný seznam 12
- nový dokument 7
- obdélník (grafický editor) 21
- oblouk 20
- obor (matematický text) 13
- obrázek 20
- obsahový objekt (SCORM) 41
- odkaz 11
- odmocnina (editor rovnic) 16
- odstavec
– formátování 10
- okno
– bibliografická databáze 32
– editor rejst íku 30
– výb r cíle k ížového odkazu 11
- okraje (pole) 17
- okraje (tabulka) 12
- operátory (editor rovnic) 13
- organizace (SCORM) 41
- otev ený tvar 21
- PDF export 50
- pole (editor rovnic) 17
- položka rejst íku 30
- popisný seznam 12
- Poznámka (t ída dokumentu) 7
- profil 46
- projekt KSMSA 5
- prom nná (matematický text) 13
- propojení
– položky indexu 30
- pr vodce
– vytvo ením nového dokumentu 7
- p esouvání 9
- rastrový obrázek 22
- rejst ík 30
– položka 30
- RichDoc framework 5
- rovnice
– íslování 18
– inline 13
- rozm r (matematický text) 13
- ádek (zobrazená rovnice) 18
- ecká písmena 13
- SCORM 41
- SCORM export 51
- ScratchPad 6
- sekce dokumentu 7
- seznam 12
- skriptovaná grafika 23
- spojka 22
- správa lokalizovaných verzí 38
- správa verzí 38
- struktura dokumentu 7
- struktury (editor rovnic) 14
- style 69
– bibliography 71
– document 71
– visual 69
- style sheet 28

suma (editor rovnic) 16	uživatelský obdélník 23
symboly (editor rovnic) 13	vektor (matematický text) 13
Systém pro podporu výuky LMS 41	vyhledání textu 44
tabulátor (editor rovnic) 18	vyplněný tvar 21
tabulka 12	WYSIWYG 5
TO-DO seznam 12	zakřivený tvar 20
transformace 25	zarovnání rovnice 18
tvar	závorky (rovnice) 16
– zakřivený 20	zlomek (editor rovnic) 16
URL 11	zobrazená rovnice 13, 17
uspořádaný seznam 12	zobrazená struktura (editor rovnic) 15
uzavřený tvar 21	

Bibliography

- [1] *DocBook.org*. <http://www.docbook.org/>.