# Systémy softvérovej dokumentácie

## Úvod do rôznych systémov generujúcich pou¾ívatetextmuskú a vývojársku dokumentáciu softvéru.

**Ondrej Holotòák**

**Ondrej Jombík**

**Systémy softvérovej dokumentácie: Úvod do rôznych systémov generujúcich pou¾ívate textmuskú a vývojársku dokumentáciu softvéru.**

Ondrej Holotòák a Ondrej Jombík

# Obsah

# Zoznam príkladov

# Kapitola 1. Úvod

Takmer ka¾dý vývojár softvéru u¾ bol niekedy v ¾ivote postavený pred problematiku dokumentácie k jeho projektu. Táto na prvý pohtextmuad banálna zále¾itos» v[1]ak v sebe skrýva viac problémov, ako by sa na prvý pohtextmuad zdalo. Softvér, ktorý neobsahuje kvalitnú obsa¾nú a vyèerpávajúcu dokumentáciu je u¾ dopredu odsúdený na zánik. V na[1]om dokumente sa vám pokúsime naèrtnú» niektoré z mo¾ných spôsobov rýchleho a efektívneho zdokumentovania softvérového produktu.

Základným faktorom, ktorý je treba ma» na zreteli, je cielová skupina. Odli[1]ný typ informácií je zaujímavý pre be¾ného pou¾ívatetextmua softvéru, iný pre vývojára aplikácie. Tomu treba následne prispôsobi» formu aj obsah dokumentácie. Preto odli[1]ujeme dva rôzne typy dokumentácie:

1. Pou¾ívatetextmuská dokumentácia

   Be¾neho pou¾ívatetextmua spravidla nezaujíma [1]truktúra softvéru ani jeho vnútorná implementácia. Nie je preòho dôle¾ité ako sú veci urobené, aké prostiedky a programovacie jazyky boli pri vývoji pou¾ité, ale ako v skutoènosti program funguje, ako sa pou¾íva a taktie¾ v akom prostredí a pod akým operaèným systémom je spustitetextmuný. Preto by sa mal dôraz klás» na podrobné, ale hlavne zrozumitetextmuné vysvetlenie v[1]etkých funkcií a mo¾ností programu.

   Je lep[1]ie, keï je pou¾ívatetextmuská dokumentácia písaná niekým iným ako samotným autorom programu. Ten toti¾ pova¾uje mnoho vecí za úplne jasné a samozrejmé, ktoré v[1]ak u¾ pre be¾ného pou¾ívatetextmua také jasné a samozrejmé nie sú. Vhodné je, keï pou¾ívatetextmuská dokumentácia obsahuje mno¾stvo príkladov a keï je celkovo písaná s vedomím, ¾e ju nebude èíta» osoba rozumejúca programovaniu.

2. Vývojárska dokumentácia

   Príslu[1]ný vývojár pracujúci na projekte potrebuje ma» zdokumentované v[1]etky vnútorné technické zále¾itosti softvéru, ako sú napr. hierarchia dediènosti tried, popis datových [1]truktúr a ich atribútov, zoznam zdrojových súborov, zoznam globálnych identifikátorov a pod. Medzi informácie takéhoto charakteru patrí tie¾ opis správneho pou¾itia príslu[1]ných vývojových nástrojov, slú¾iacich k zostavovaniu programu.

   Základným úèelom existencie vývojárskej dokumentácie je údr¾ba a roz[1]íritetextmunos» softvérového produktu. Nutné je zdokumentova» najmä v[1]etky ne[1]tandartné programovacie kon[1]trukcie a taktie¾ podrobný popis postupu ladenia programu.

Z prechádzajúcich definícií je teda zrejmé, ¾e tieto dva typy dokumentácie sú úplne odli[1]né. Nie je mo¾né, aby dokumentácia prvého typu nahrádzala druhú alebo opaène. U vývojárskej dokumentácie platí pravidlo písa» ju priamo za behu vytvárania programu. Ak sa vnútorná implementácia alebo [1]truktúra zmení, zmení sa aj príslu[1]ná èas» dokumentácie. Pou¾ívatetextmuskú dokumentáciu je vhodné taktie¾ zaèa» písa» èo najskôr, ale a¾ po dosiahnutí aspoò základnej funkcionality programu.

Oba typy dokumetácie pí[1]u iní textmuudia a tomu sú aj prispôsobené nástroje na ich tvorbu a údr¾bu. Ich pou¾itie bude opísané v nasledujúcich kapitolách tohto dokumentu.

# Kapitola 2. Systémy pou¾ívatetextmuskej dokumentácie softvéru

Najznámej[1]ím a najpou¾ívanej[1]ím systémom pre tvorbu pou¾ívatetextmuskej dokumentácie je DocBook (http://www.oasis-open.org/docbook/). Tento otvorený nástroj postavený nad otvorenými technológiami ponúka systém písania [1]truktúrových dokumentov pou¾itím SGML alebo XML. ©peciálne je predurèený na písanie kníh zaoberajúcich sa poèítaèovým hardvérom a softvérom, ale rozhodne nie je limitovaný len na túto oblas». DocBook je SGML definícia typu dokumentu (Document Type Definition - DTD).

Mnoho známych firiem ako napr. TcX (tvorcovia databázového systému MySQL) alebo TrollTech (tvorcovia grafickej kni¾nice QT2) pou¾íva pre dokumentovanie svojich produktov vlastné nástroje. Odhliadnuc od ostatných faktorov mô¾e by» výsledkom pou¾itia vlastného dokumentaèného systému pomerne »a¾ké a komplikované zlep[1]ovanie a roz[1]irovanie dokumentácie od rôznych prispievatetextmuov. Preto sa pou¾ívanie otvorených technológií v procese vytvárania dokumentácie pova¾uje za vysoko ¾iadúce a silne produktívne.

DocBook dokument je písaný pou¾itím tzv. znaèkovacích jazykov (markup languages) ako je SGML a XML. DocBook existuje v oboch týchto verziách. Podobne ako pri HTML dokumentoch sa [1]pecálnymi znaèkami (tagmi) oznaèujú jednotlivé miesta alebo elementy. V prípade HTML dokumentu sa v[1]ak pomocou znaèiek definuje dizajn a výzor výsledného dokumentu (zmena farby, zmena písma, zvýraznenie a pod.). V DocBook dokumente neexistuje ¾iadna znaèka manipulujúca priamo s dizajnom. Dokument je [1]truktúrovaný *logicky*, èi¾e oznaèované sú èasti s rovnakým významom (napr. mená súborov, premenných, produktov, textmuudí, atï.). Výsledný dizajn je komponovaný pomocou jazyka DSSSL (http://www.jclark.com/dsssl/) (Document Style Semantics and Specification Language), ten v[1]ak nie je pre be¾ného pou¾ívatetextmua podstatný.

Rozdiely medzi XML verziou a SGML verziou DocBook sú minimálne. Okrem odli[1]nej hlavièky dokumentu a rôznej implementácii asi dvoch tagov sú obe verzie úplne identické. Podstatný rozdiel je v samotnom spracovaní dokumentu. Pri SGML verzii sa pou¾íva Jade Wrapper (jw) a pri XML verzii nástroje postavené nad kni¾nicou libxml2. My vo svojich príkladoch budeme pou¾íva» SGML verziu.

Základný dokument vyzerá nasledovne:

**Príklad 2-1. Základný DocBook SGML dokument**

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V3.1//EN" [ ]>
<book lang="sk">
<bookinfo>
 <date>2002-12-09</date>
 <title>Môj prvý DocBook dokument</title>
 <subtitle>s týmto podnadpisom</subtitle>
</bookinfo>
```

```
<chapter>
<title>Nadpis prvej kapitoly</title>

<para>Text prvej kapitoly.</para>

</chapter>
</book>
```

DocBook dokument (<book>) sa delí na kapitoly (<chapter>), sekcie v kapitolách (<sect1>) a podsekcie (<sect2>). Podsekcie mo¾u ma» ïal[1]ie podsekcie (<sect3>, <sect4>). Kapitoly je mo¾né zdru¾ova» pomocou tagu <part>. Podobne viac kníh je mo¾né spoji» do kompaktného celku pomocou tagu <set>.

**Príklad 2-2. Základné delenie DocBook SGML dokumentu**

```
<chapter><title>Nadpis prvej kapitoly</title>
 <sect1><title>Prvá sekcia v prvej kapitole</title>>
  <sect2><title>Prvá podsekcia v prvej sekcii</title>
   <para>Text. Text. Text. Text.</para>
  </sect2>
  <sect2><title>Druhá podsekcia v prvej sekcii</title>
  </sect2>
 </sect1>
 <sect1 ...>
  <para> ... </para>
 </sect1>
</chapter>
```

Ako u¾ bolo spomenuté, DocBook dokument má logickú [1]truktúru podtextmua významu. Znaèky sa pou¾ívajú na oznaèovanie výrazov alebo slov rovnakého typu. Oznaèujú sa teda napríklad mená súborov, premenných, kon[1]tánt, funkcií, príkazov, produktov. Ïalej tie¾ e-mailové adresy alebo odkazy na webstránky. Je mo¾né tie¾ vytvára» zoznamy alebo tabutextmuky, vklada» obrázky, atï.

**Príklad 2-3. Uká¾ka niektorých DocBook SGML tagov**

```
<filename>/etc/passwd</filename>
<filename class=headerfile>stdio.h</filename>
<varname>$selected_user</varname>
<constant>BUFFER_SIZE</constant>
<function>proctable_init()</function>
<command>grep</command>
<application>jade</application>
<productname>DocBook</productname>
<email>platon@pobox.sk</email>
<ulink url="http://www.platon.sk/">Platon SDG</>
```

Na poslednom príklade je vidie» mo¾nos» skracovania jednoduchých SGML tagov pomocou ukonèovacej znaèky </>.

Medzi najvýznamnej[1]ie výhody DocBook dokumentácie patrí aj vetextmuký poèet mo¾ných formátov výstupných dokumentov. Sú to najmä HTML, RTF, PDF, PS, TeX, èistý text a manuálové stránky. Generovanie prebieha pomocou programu jade (predpokladáme vygenerovanie súboru `myfile.html` zo súboru `myfile.sgml`).

```
$ jade -d /usr/lib/sgml/stylesheets/dbtohtml.dsl -t sgml myfile.sgml > myfile.htm
```

Nie je nutné pou¾íva» priamo analyzátor jade. Existujú u¾ predpripravené *shell* skripty pre generovanie konkrétnych typov výstupných súborov. Sú to napr. db2html, db2pdf, ale aj docbook2tex èi docbook2txt. Tak¾e pre vygenerovanie HTML dokumentácie staèí pou¾i» nasledovný príkaz.

```
$ db2html myfile.sgml
```

Vzhtextmuad výsledného HTML dokumentu sa jednoducho prispôsobuje pou¾itím CSS (Cascading Style Sheets).

# Kapitola 3. Systémy vývojárskej dokumentácie softvéru

Pre generovanie vývojárskej dokumentácie existuje viacero nástrojov v závislosti na tom, v akom programovacom jazyku je aplikácia napísaná. Pre C/C++ je to Doxygen (http://www.doxygen.org/), ktorý sa vyvinul z Doc++ (http://docpp.sourceforge.net/). Pre programovací jazyk PHP poznáme hneï niekotextmuko nástrojov: phpDoc (http://www.phpdoc.de/), phpDocumentor (http://www.phpdoc.org/), PearDoc a PearDoc2 (http://pear.php.net/manual/). Posledný z menovaných má pred sebou zrejme najlep[1]iu budúcnos», nakotextmuko je aktívne vyvíjaný a zrejme sa stane hlavným dokumentaèným nástrojom pre PHP. Nakoniec pre programovací jazyk Java je to známy JavaDoc (http://java.sun.com/j2se/javadoc/).

Vývojárska dokumentácia sa pí[1]e priamo do zdrojových kódov programu v podobe [1]peciálnych komentárov. Dokumentujú sa v[1]etky entity programovacieho jazyka: triedy a ich metódy i atribúty, globálne a lokálne funkcie i premenné, kon[1]tanty, dátové typy ako sú napr. [1]truktúry, výètové typy a iné. Dokumentaèný systém pomocou [1]peciálnych znaèiek rozpozná typ informácie a na základe toho ju zakomponuje do výslednej dokumentácie.

Nespornými výhodami uvedeného prístupu sú najmä u[1]etrný èas a komplexnos». Vïaka tomu, ¾e sa dokumentácia pí[1]e priamo do zdrojovýh súborov, nie je nutné pracova» so zdrojovým kódom a dokumentaèným súborom súèasne. Ako u¾ bolo spomenuté, vývojársku dokumentáciu je vhodné písa» priamo poèas vývoja aplikácie. A tak napríklad po pridaní novej metódy sa k nej zároveò napí[1]e aj jednoduchý komentár, ktorý je jednak pou¾itetextmuný pre informovanie o funkènosti metódy priamo v zdrojovom kóde a samozrejme bude tie¾ pou¾itý na vygenerovanie vývojárskej dokumentácie. Klasické komentáre je mo¾né v zdrojovom kóde pou¾íva» aj naïalej, do dokumentácie zahrnuté nebudú.

Treba zdôrazni», ¾e dokumentaèné systémy sú schopné vetextmuké mno¾stvo poznatkov odvodi» priamo zo zdrojového kódu, keï¾e v sebe spravidla obsahujú analyzátory daného programovacieho jazyka. Napríklad do dokumentaèného komentáru ku funkcii, metóde alebo triede nie je nutné písa» jej meno, keï¾e to sa dá odvodi» z príslu[1]nej deklarácie. Toto je aj hlavný dôvod, preèo pre ka¾dý programovací jazyk existujú odli[1]né dokumentaèné nástroje vývojárskej dokumentácie, napriek tomu, ¾e ich pou¾itie je v zásade toto¾né.

Uvádzame uká¾ky zdokumentovania jednoduchej triedy v troch odli[1]ných programovacích jazykoch (C++, PHP, Java) pomocou ich dokumentaèných systémov (Doxygen, PearDoc, JavaDoc). Zameriame sa najmä na spoloèné èrty, ktoré su pre v[1]etky systémy zhodné. Zaèíname príkladom pre Doxygen.

**Príklad 3-1. Zdokumentovaný C++ zdrojový kód pre Doxygen**

```
/**
 * Example class
 *
```

```
 * More elaborate class description.
 */
class Example
{
 public:

 /**
  * Example object constructor
  *
  * More elaborate description of the constructor.
  *
  * @param    bar    a bar constructor parameter
  * @param    foo    a foo constructor parameter
  * @return   void
  */
 Example(int bar = 0, const char *foo = "foo");

 /**
  * Example object destructor
  *
  * More elaborate description of the destructor.
  */
 ~Example();
};
```

**Príklad 3-2. Zdokumentovaný PHP zdrojový kód pre PearDoc**

```
/**
 * Example class
 *
 * More elaborate class description.
 */
class Example
{
 /**
  * Example object constructor
  *
  * More elaborate description of the constructor.
  *
  * @param    int      bar   a bar constructor parameter
  * @param    string   foo   a foo constructor parameter
  * @return   boolean        true if this example is right,
  *                          false otherwise
  */
 function Example($bar = 0, $foo = 'foo')
 {
  // Constructor code goes here. Note that PHP does not
  // have anything like header files with declarations.
  return true;
 }
}
```

**Príklad 3-3. Zdokumentovaný Java zdrojový kód pre JavaDoc**

```
/**
 * Example class
 *
 * More elaborate class description.
 */
class Example
{
 /**
  * Example object constructor
  *
  * More elaborate description of the constructor.
  *
  * @param    bar    a bar constructor parameter
  * @param    foo    a foo constructor parameter
  * @return   void
  */
 Example(int bar = 0, const char *foo = "foo")
 {
  // Constructor code goes here
 }
}
```

Z príkladov vidie», ¾e [1]peciálne komentáre zaèínajú sekvenciou /**. Klasické /* komentáre sa ignorujú. Boli pou¾ité znaèky @param pre definovanie vstupného funkèného parametru a @return pre definovanie navratovej hodnoty z funkcie. V[1]imnite si, ¾e zdrojový kód C++ u¾ obsahuje informáciu o dátovom type vstupného parametra, preto sa v komentári nevyskytuje. Dokumentaèný systém Doxygen si ju zistí sám. Naproti tomu zdrojový kód PHP túto informáciu v sebe nemá, tak¾e musí by» v dokumentaènom komentári uvedená.

Medzi ïal[1]ie znaèky pou¾ívané na dokumentovanie funkcií alebo metód patria okrem @param a @return tie¾ @access vyjadrujúci prístupnos» metódy (public, private, ...) a @retval pou¾ívajúci sa na opis návratovej hodnoty parametru funkcie definovaného referenciou (adresou). Ïal[1]ie v[1]estranne pou¾itetextmuné znaèky sú:

```
@author      -- autor entity (triedy, metódy, funkcie)
@version     -- verzia entity
@date        -- dátum vzniku
@since       -- minimálna verzia majúca nejakú funkcionalitu
@deprecated  -- stará a u¾ nepodporovaná funkcionalita
@see         -- odkaz na súvisiacu entitu
@todo        -- popis nedokonèných vymo¾eností
```

**Príklad 3-4. Pou¾itie ïal[1]ích dokumentaèných znaèiek**

```
/**
```

```
 * Extensive Example class
 *
 * This extensive class does some extensive job.
 *
 * @author     Ondrej Jombik <nepto@pobox.sk>
 * @author     Ondrej Holotnak <beast@host.sk>
 * @version    1.0
 * @date       2002-12-19
 */
class Extensive_Example
{
 /**
  * Normal method doing something
  *
  * @param   some    some integer argument
  * @retval  thing   allocated string filled with thing
  * @return          test result
  * @author  Someone External <someone@example.com>
  * @see     Test()
  * @see     ~Test()
  * @todo    make this method something doing
  */
 int doSomething(int some, char **thing);
};
```

Podobne ako pri pou¾ívatetextmuskej dokumentácii je aj tu mo¾nos» vygenerova» finálnu dokumentáciu vo viacerých formátoch. Aplikácia Doxygen podporuje nasledovné výstupné formáty: HTML, LaTeX, RTF, XML a manuálové stránky. V[1]etko sa konfiguruje pomocou prehtextmuadného konfiguraèného súboru `Doxyfile`. Pokial spustíte program **doxygen** s prepínaèom `-g`, základný konfiguraèný súbor bude zapísaný na disk. Obsahuje vetextmua komentárov, tak¾e jeho modifikácia je jednoduchá a priamoèiara. Samotná generácia dokumentácie prebieha nasledovným príkazom:

```
$ doxygen
```

Ostatné vymenované nástroje sa pou¾ívajú obdobne. Nemusia v[1]ak obsahova» podporu v[1]etkých vymenovaných formátov. V[1]etky v[1]ak urèite obsahujú mo¾nos» generovania HTML výstupu, ktorý sa pova¾uje za najdôle¾itej[1]í a najpou¾itetextmunej[1]í.

# Kapitola 4. Iné systémy pre dokumentáciu softvéru

Existujú aj ïal[1]ie systémy pre dokumentáciu softvéru. O ¾iadnom z týchto systémov sa nedá poveda», èi patrí do skupiny pou¾ívatetextmuskej alebo vývojárskej dokumentácie. Ka¾dý z nich v dobe svojho vzniku sledoval svoj konkrétny cietextmu a bol takpovediac [1]itý na mieru ¾iadanému a predpokladanému úèelu. Sú to napríklad:

- PerlDoc
- man
- info

PerlDoc (http://www.perldoc.com/) je dokumentaèný systém pou¾ívaný komunitou okolo skriptovacieho jazyka Perl (http://www.perl.org/). Jeho vetextmukému rozvoju dopomohla najmä existencia CPAN (http://www.cpan.org/) - archívu znovupou¾itetextmuného kódu pre Perl. Dokumentácia sa pí[1]e priamo do zdrojových kódov pomocou [1]peciálnych POD znaèiek. Pou¾ívatelia si ju potom prezerajú pomocou systémového Perl skriptu **perldoc**. Existuje mnoho konvertorov PerlDoc dokumentácie do rôznych výstupných formátoch, napr. HTML.

Dokumentaèný systém systémových manuálových stránok (man stránky) vznikol e[1]te v èasoch vzniku operaèného systému UNIX a jej dodnes aktívne vyvíjaný. Pre formátovanie výstupu sa pou¾íva program nroff a stránky sa prezerajú príkazom **man**. Vyskytujú sa takmer na ka¾dom UNIX alebo GNU/Linux systéme. Na disku je ulo¾ená ka¾dá man stránka osobitne a zkomprimovaná. Taktie¾ existujú nástroje na ich konverziu do iných formátov.

Výsledkom hnutia GNU (GNU is not UNIX) je info dokumentácia, ktorá sa sna¾í nahradi» pou¾ívané manuálové stránky. Tento dokumentaèný systém sa v[1]ak kvoli komplikovanému ovládaniu a nevábnemu vzhtextmuadu neujal v takej miere ako si jeho tvorcovia predstavovali. Väè[1]ina GNU nástrojov v[1]ak obsahuje obsiahlej[1]iu a podrobnej[1]iu dokumentáciu vo svojich info stránkach ako v prísl[1]ných man stránkach. Stránky sa prezerajú príkazom **info**.