

Systemy softvérovej dokumentácie

**Úvod do rôznych systémov generujúcich
používateľskú a vývojársku dokumentáciu
softvéru.**

Ondrej Holotňák

Ondrej Jombík

Systémy softvérovej dokumentácie: Úvod do rôznych systémov generujúcich používateľskú a vývojársku dokumentáciu softvéru.

Ondrej Holotňák a Ondrej Jombík

Copyright © 2002, 2003, 2004 Platon Software Development Group (<http://platon.sk/>)

Obsah

1. Úvod	1
2. Systémy používateľskej dokumentácie softvéru	3
3. Systémy vývojárskej dokumentácie softvéru	6
4. Iné systémy pre dokumentáciu softvéru	10

Zoznam príkladov

2-1. Základný DocBook SGML dokument	3
2-2. Základné delenie DocBook SGML dokumentu	4
2-3. Ukážka niektorých DocBook SGML tagov	4
3-1. Zdokumentovaný C++ zdrojový kód pre Doxygen.....	6
3-2. Zdokumentovaný PHP zdrojový kód pre PearDoc.....	7
3-3. Zdokumentovaný Java zdrojový kód pre JavaDoc	7
3-4. Použitie ďalších dokumentačných značiek	8

Kapitola 1. Úvod

Takmer každý vývojár softvéru už bol niekedy v živote postavený pred problematiku dokumentácie k jeho projektu. Táto na prvý pohľad banálna záležitosť však v sebe skrýva viac problémov, ako by sa na prvý pohľad zdalo. Softvér, ktorý neobsahuje kvalitnú obsažnú a vyčerpávajúcu dokumentáciu je už dopredu odsúdený na zánik. V našom dokumente sa vám pokúsime načrtnúť niektoré z možných spôsobov rýchleho a efektívneho zdokumentovania softvérového produktu.

Základným faktorom, ktorý je treba mať na zreteli, je cieľová skupina. Odlišný typ informácií je zaujímavý pre bežného používateľa softvéru, iný pre vývojára aplikácie. Tomu treba následne prispôbiť formu aj obsah dokumentácie. Preto odlišujeme dva rôzne typy dokumentácie:

1. Používateľská dokumentácia

Bežného používateľa spravidla nezaujíma štruktúra softvéru ani jeho vnútorná implementácia. Nie je preňho dôležité ako sú veci urobené, aké prostriedky a programovacie jazyky boli pri vývoji použité, ale ako v skutočnosti program funguje, ako sa používa a taktiež v akom prostredí a pod akým operačným systémom je spustiteľný. Preto by sa mal dôraz klásť na podrobné, ale hlavne zrozumiteľné vysvetlenie všetkých funkcií a možností programu.

Je lepšie, keď je používateľská dokumentácia písaná niekým iným ako samotným autorom programu. Ten totiž považuje mnoho vecí za úplne jasné a samozrejmé, ktoré však už pre bežného používateľa také jasné a samozrejmé nie sú. Vhodné je, keď používateľská dokumentácia obsahuje množstvo príkladov a keď je celkovo písaná s vedomím, že ju nebude čítať osoba rozumejúca programovaniu.

2. Vývojárska dokumentácia

Príslušný vývojár pracujúci na projekte potrebuje mať zdokumentované všetky vnútorné technické záležitosti softvéru, ako sú napr. hierarchia dedičnosti tried, popis datových štruktúr a ich atribútov, zoznam zdrojových súborov, zoznam globálnych identifikátorov a pod. Medzi informácie takéhoto charakteru patrí tiež opis správneho použitia príslušných vývojových nástrojov, slúžiacich k zostavovaniu programu.

Základným účelom existencie vývojárskej dokumentácie je údržba a rozšíriteľnosť softvérového produktu. Nutné je zdokumentovať najmä všetky neštandardné programovacie konštrukcie a taktiež podrobný popis postupu ladenia programu.

Z prechádzajúcich definícií je teda zrejmé, že tieto dva typy dokumentácie sú úplne odlišné. Nie je možné, aby dokumentácia prvého typu nahrádzala druhú alebo opačne. U vývojárskej dokumentácie platí pravidlo písať ju priamo za behu vytvárania programu. Ak sa vnútorná implementácia alebo štruktúra zmení, zmení sa aj príslušná časť dokumentácie. Používateľskú dokumentáciu je vhodné taktiež začať písať čo najskôr, ale až po dosiahnutí aspoň základnej funkcionality programu.

Oba typy dokumentácie píše iní ľudia a tomu sú aj prispôsobené nástroje na ich tvorbu a údržbu. Ich použitie bude opísané v nasledujúcich kapitolách tohto dokumentu.

Kapitola 2. Systémy používateľskej dokumentácie softvéru

Najznámejším a najpoužívanejším systémom pre tvorbu používateľskej dokumentácie je DocBook (<http://www.oasis-open.org/docbook/>). Tento otvorený nástroj postavený nad otvorenými technológiami ponúka systém písania štruktúrových dokumentov použitím SGML alebo XML. Špeciálne je predurčený na písanie kníh zaoberajúcich sa počítačovým hardvérom a softvérom, ale rozhodne nie je limitovaný len na túto oblasť. DocBook je SGML definícia typu dokumentu (Document Type Definition - DTD).

Mnoho známych firiem ako napr. TcX (tvorcovia databázového systému MySQL) alebo TrollTech (tvorcovia grafickej knižnice QT2) používa pre dokumentovanie svojich produktov vlastné nástroje. Odhliadnuc od ostatných faktorov môže byť výsledkom použitia vlastného dokumentačného systému pomerne ťažké a komplikované zlepšovanie a rozširovanie dokumentácie od rôznych prispievateľov. Preto sa používanie otvorených technológií v procese vytvárania dokumentácie považuje za vysoko žiadúce a silne produktívne.

DocBook dokument je písaný použitím tzv. značkovacích jazykov (markup languages) ako je SGML a XML. DocBook existuje v oboch týchto verziách. Podobne ako pri HTML dokumentoch sa špeciálnymi značkami (tagmi) označujú jednotlivé miesta alebo elementy. V prípade HTML dokumentu sa však pomocou značiek definuje dizajn a výzor výsledného dokumentu (zmena farby, zmena písma, zvýraznenie a pod.). V DocBook dokumente neexistuje žiadna značka manipulujúca priamo s dizajnom. Dokument je štruktúrovaný *logicky*, čiže označované sú časti s rovnakým významom (napr. mená súborov, premenných, produktov, ľudí, atď.). Výsledný dizajn je komponovaný pomocou jazyka DSSSL (<http://www.jclark.com/dsssl/>) (Document Style Semantics and Specification Language), ten však nie je pre bežného používateľa podstatný.

Rozdiely medzi XML verziou a SGML verziou DocBook sú minimálne. Okrem odlišnej hlavičky dokumentu a rôznej implementácii asi dvoch tagov sú obe verzie úplne identické. Podstatný rozdiel je v samotnom spracovaní dokumentu. Pri SGML verzii sa používa Jade Wrapper (jw) a pri XML verzii nástroje postavené nad knižnicou libxml2. My vo svojich príkladoch budeme používať 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 môžu mať ďalšie podsekcie (<sect3>, <sect4>). Kapitoly je možné združovať pomocou tagu <part>. Podobne viac kníh je možné spojiť 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 už bolo spomenuté, DocBook dokument má logickú štruktúru podľa významu. Značky sa používajú na označovanie výrazov alebo slov rovnakého typu. Označujú sa teda napríklad mená súborov, premenných, konštánt, funkcií, príkazov, produktov. Ďalej tiež e-mailové adresy alebo odkazy na webstránky. Je možné tiež vytvárať zoznamy alebo tabuľky, vkladať obrázky, atď.

Príklad 2-3. Ukážka 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 vidieť možnosť skracovania jednoduchých SGML tagov pomocou ukončovacej značky `</>`.

Medzi najvýznamnejšie výhody DocBook dokumentácie patrí aj veľký počet možných formátov výstupných dokumentov. Sú to najmä HTML, RTF, PDF, PS, TeX, čistý 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 používať priamo analyzátor `jade`. Existujú už predpripravené *shell* skripty pre generovanie konkrétnych typov výstupných súborov. Sú to napr. `db2html`, `db2pdf`, ale aj `docbook2tex` či `docbook2txt`. Takže pre vygenerovanie HTML dokumentácie stačí použiť nasledovný príkaz.

```
$ db2html myfile.sgml
```

Vzhľad výsledného HTML dokumentu sa jednoducho prispôsobuje použitím 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 hneď niekoľko 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 najlepšiu budúcnosť, nakoľko je aktívne vyvíjaný a zrejme sa stane hlavným dokumentačným nástrojom pre PHP. Nakoniec pre programovací jazyk Java je to známy JavaDoc (<http://java.sun.com/j2se/javadoc/>).

Vývojárska dokumentácia sa píše priamo do zdrojových kódov programu v podobe špeciálnych komentárov. Dokumentujú sa všetky entity programovacieho jazyka: triedy a ich metódy i atribúty, globálne a lokálne funkcie i premenné, konštanty, dátové typy ako sú napr. štruktúry, výčtové typy a iné. Dokumentačný systém pomocou špeciálnych značiek rozpozná typ informácie a na základe toho ju zakomponuje do výslednej dokumentácie.

Nespornými výhodami uvedeného prístupu sú najmä ušetrný čas a komplexnosť. Vďaka tomu, že sa dokumentácia píše priamo do zdrojových súborov, nie je nutné pracovať so zdrojovým kódom a dokumentačným súborom súčasne. Ako už bolo spomenuté, vývojársku dokumentáciu je vhodné písať priamo počas vývoja aplikácie. A tak napríklad po pridaní novej metódy sa k nej zároveň napíše aj jednoduchý komentár, ktorý je jednak použiteľný pre informovanie o funkčnosti metódy priamo v zdrojovom kóde a samozrejme bude tiež použitý na vygenerovanie vývojárskej dokumentácie. Klasické komentáre je možné v zdrojovom kóde používať aj naďalej, do dokumentácie zahrnuté nebudú.

Treba zdôrazniť, že dokumentačné systémy sú schopné veľké množstvo poznatkov odvodiť priamo zo zdrojového kódu, keďže v sebe spravidla obsahujú analyzátory daného programovacieho jazyka. Napríklad do dokumentačného komentáru ku funkcii, metóde alebo triede nie je nutné písať jej meno, keďže to sa dá odvodiť z príslušnej deklarácie. Toto je aj hlavný dôvod, prečo pre každý programovací jazyk existujú odlišné dokumentačné nástroje vývojárskej dokumentácie, napriek tomu, že ich použitie je v zásade totožné.

Uvádzame ukážky zdokumentovania jednoduchej triedy v troch odlišných programovacích jazykoch (C++, PHP, Java) pomocou ich dokumentačných systémov (Doxygen, PearDoc, JavaDoc). Zameriame sa najmä na spoločné črty, ktoré su pre všetky systémy zhodné. Začíname 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 vidieť, že špeciálne komentáre začínajú sekvenciou `/**`. Klasické `/*` komentáre sa ignorujú. Boli použité značky `@param` pre definovanie vstupného funkčného parametru a `@return` pre definovanie navratovej hodnoty z funkcie. Všimnite si, že zdrojový kód C++ už obsahuje informáciu o dátovom type vstupného parametru, preto sa v komentári nevyskytuje. Dokumentačný systém Doxygen si ju zistí sám. Naproti tomu zdrojový kód PHP túto informáciu v sebe nemá, takže musí byť v dokumentačnom komentári uvedená.

Medzi ďalšie značky používané na dokumentovanie funkcií alebo metód patria okrem `@param` a `@return` tiež `@access` vyjadrujúci prístupnosť metódy (`public`, `private`, ...) a `@retval` používajúci sa na opis návratovej hodnoty parametru funkcie definovaného referenciou (adresou). Ďalšie všestranne použiteľné značky sú:

```

@author    -- autor entity (triedy, metódy, funkcie)
@version   -- verzia entity
@date     -- dátum vzniku
@since    -- minimálna verzia majúca nejakú funkcionálnosť
@deprecated -- stará a už nepodporovaná funkcionálnosť
@see      -- odkaz na súvisiacu entitu
@todo     -- popis nedokončených vymožeností

```

Príklad 3-4. Použitie ďalších dokumentačných značiek

```

/**

```

```

* Extensive Example class
*
* This extensive class does some extensive job.
*
* @author      Ondrej Jombik <nepto@pobox.sk>
* @author      Ondrej Holotnak <beast@host.sk>
* @version     1.2
* @date        2004-03-26
*/
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 používateľskej dokumentácii je aj tu možnosť vygenerovať finálnu dokumentáciu vo viacerých formátoch. Aplikácia Doxygen podporuje nasledovné výstupné formáty: HTML, LaTeX, RTF, XML a manuálové stránky. Všetko sa konfiguruje pomocou prehľadného konfiguračného súboru Doxyfile. Pokiaľ spustíte program **doxygen** s prepínačom `-g`, základný konfiguračný súbor bude zapísaný na disk. Obsahuje veľa komentárov, takže jeho modifikácia je jednoduchá a priamočiara. Samotná generácia dokumentácie prebieha nasledovným príkazom:

\$ **doxygen**

Ostatné vymenované nástroje sa používajú obdobne. Nemusia však obsahovať podporu všetkých vymenovaných formátov. Všetky však určite obsahujú možnosť generovania HTML výstupu, ktorý sa považuje za najdôležitejší a najpoužiteľnejší.

Kapitola 4. Iné systémy pre dokumentáciu softvéru

Existujú aj ďalšie systémy pre dokumentáciu softvéru. O žiadnom z týchto systémov sa nedá povedať, či patrí do skupiny používateľskej alebo vývojárskej dokumentácie. Každý z nich v dobe svojho vzniku sledoval svoj konkrétny cieľ a bol takpovediac šitý na mieru žiadanému a predpokladanému účelu. Sú to napríklad:

- PerlDoc
- man
- info

PerlDoc (<http://www.perldoc.com/>) je dokumentačný systém používaný komunitou okolo skriptovacieho jazyka Perl (<http://www.perl.org/>). Jeho veľkému rozvoju dopomohla najmä existencia CPAN (<http://www.cpan.org/>) - archívu znovupoužiteľného kódu pre Perl. Dokumentácia sa píše priamo do zdrojových kódov pomocou špeciálnych POD značiek. Používatelia si ju potom prezerajú pomocou systémového Perl skriptu **perldoc**. Existuje mnoho konvertorov PerlDoc dokumentácie do rôznych výstupných formátov, napr. HTML.

Dokumentačný systém systémových manuálových stránok (man stránky) vznikol ešte v časech vzniku operačného systému UNIX a jej dodnes aktívne vyvíjaný. Pre formátovanie výstupu sa používa program **nroff** a stránky sa prezerajú príkazom **man**. Vyskytujú sa takmer na každom UNIX alebo GNU/Linux systéme. Na disku je uložená každá man stránka osobitne a zkomprimovaná. Taktiež existujú nástroje na ich konverziu do iných formátov.

Výsledkom hnutia GNU (GNU is not UNIX) je info dokumentácia, ktorá sa snaží nahradiť používané manuálové stránky. Tento dokumentačný systém sa však kvôli komplikovanému ovládaniu a nevábnemu vzhľadu neujal v takej miere ako si jeho tvorcovia predstavovali. Väčšina GNU nástrojov však obsahuje obsiahlejšiu a podrobnejšiu dokumentáciu vo svojich info stránkach ako v príslušných man stránkach. Stránky sa prezerajú príkazom **info**.