Dokumentační projekt wxXmlSerializer

Documentation project of wxXmlSerializer

Václav Kvasnička

Bakalářská práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Václav KVASNIČKA**
Osobní číslo:	**A07057**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Téma práce:	**Dokumentační projekt mupliplatformní open-source SW knihovny wxXmlSerializer**

Zásady pro vypracování:

1. **Vytvořte literární rešerši na téma aktuálně dostupné open-source SW knihovny vhodné pro zajištění datové persistence objektů tříd jazyka C++. Srovnejte je z hlediska programových vlastností, funkcionality a platformní závislosti.**
2. **Pomocí systému Doxygen vytvořte sadu elektronických dokumentů obsahujících popis API knihovny. Rozšiřte existující nápovědu o stručné příklady využití knihovny.**
3. **Vytvořte sadu elektronických dokumentů demonstrujících způsob práce s SW knihovnou (tvorba serializovatelných tříd, serializace/deserializace jednotlivých objektů, seznamů objektů a stromů objektů, tvorba vlastních serializovatelných datových typů, ...)**
4. **Vytvořte programovou a uživatelskou dokumentaci existujících vzorových projektů knihovny wxXmlSerializer.**
5. **Vytvořte HTML stránky obsahující výše uvedené elektronické dokumenty a publikujte je na SourceForge.net (http://sourceforge.net/projects/wxxs/)**
6. **Aktivujte a nakonfigurujte Wiki stránky na SourceForge.net (http://sourceforge.net/projects/wxxs/)**
7. **Veškerá vytvořená dokumentace bude v anglickém jazyce.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **BLIŽŇÁK, Michal. Systémové programování. 1. vyd. Zlín: UTB ve Zlíně, 2005. 202 s. ISBN 80-7318-364-1.**
2. **BLIŽŇÁK, M, DULÍK, T, VAŠEK, V, A Persistent Cross-Platform XML-based Class Objects Container, In Proceedings of the 10th WSEAS International Conference on AUTOMATION & INFORMATION, Prague, Czech Republic, WSEAS Press, 2009, p. 316-321, ISBN 978-960-474-064-2, ISSN 1790-5117.**
3. **BLIŽŇÁK, M, DULÍK, T, VAŠEK, V, A Persistent Cross-Platform Class Objects Container for C++ and wxWidgets , WSEAS TRANSACTIONS on COMPUTERS, Issue 5, Volume 8, May 2009, p.778-787, ISSN 1109-2750.**
4. **SMART, Julian, HOCK, Kevin. Cross-Platform GUI Programming with wxWidgets, Prentice Hall, 2006, ISBN 0-13-147381-6.**
5. **PRATA, Stephen. Mistrovství v C++. 3. vydání. Brno: Computer Press, 2007. 1120 s. ISBN 978-80-251-1749-1.**
6. **KOSEK, Jiří. XML pro každého - podrobný průvodce. 1. vydání. Grada Publishing, 2000. 164 s. ISBN 80-7169-860-1.**
7. **KOSEK, Jiří. HTML - tvorba dokonalých WWW stránek - podrobný průvodce. 1. vydání. Praha: Grada Publishing, 1998. 296 s. ISBN 80-7169-608-0.**
8. **wxXmlSerializer at SourceForge.net, URL: http://sourceforge.net/projects/wxxs**
9. **wxWidgets Documentation, URL: http://www.wxwidgets.org/docs/**
10. **Doxygen Manual, URL: http://www.doxygen.nl/manual.html**

Vedoucí bakalářské práce: **Ing. Michal Bližňák, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **5. března 2010**

Termín odevzdání bakalářské práce: **1. června 2010**

Ve Zlíně dne 5. března 2010

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této bakalářské práce bylo vytvořit sadu elektronických dokumentů k open-source knihovně wxXmlSerializer. Tyto elektronické dokumenty popisují API knihovny, základní způsob práce s ní a již existující vzorové projekty využívající knihovnu.

Klíčová slova:

Datová persitence, serializace dat, wxWidgets, wxXmlSerializer, C++

ABSTRACT

The aim of this bachelor thesis was to create a set of the electronical documents for open-source software library wxXmlSerializer. These electronical documents contain description of the API of the library, description of how to work with the library and description of existing sample projects.

Keywords:

Data persistence, serialization, wxWidgets, wxXmlSerializer, C++

Děkuji mému vedoucímu, Ing. Michalu Bližňákovi, Ph.D., za odbornou podporu při vypracování mé bakalářské práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….…….

podpis diplomanta

OBSAH

ÚVOD

Cílem této práce bylo vytvořit sadu elektronických dokumentů popisujících wxXmlSerializer. Tato open-source multiplatformní knihovna poskytuje funkcionality pomocí kterých lze perzistentně ukládat hierarchicky uspořádané instance tříd jazyka C++.

Práce je rozdělena do dvou částí. V teoretické části byla provedena literární rešerše na téma aktuálně dostupných open-source knihoven vhodných pro zajištění datové persistence objektů tříd jazyka C++.

Praktická část práce je zaměřena především na tvorbu dokumentace ke knihovně. Je zde popsána práce s dokumentačním systémem Doxygen dále je zde také popis webových stránek, na kterých je veškerá dokumentace zpřístupněna. Na závěr jsou popsány stránky Wiki knihovny wxXmlSerializer.

I. TEORETICKÁ ČÁST

1 OPEN-SOURCE SW KNIHOVNY VHODNÉ PRO ZAJIŠTĚNÍ DATOVÉ PERZISTENCE OBJEKTŮ JAZYKA C++

1.1 Co je datová perzistence

Perzistentně uložená data jsou taková data, jejichž existence je zaručena i po ukončení programu, který s těmito daty pracoval nebo je vytvořil.

S pojmem datová perzistence souvisí i pojem serializace. Serializací rozumíme proces, při kterém jsou datové struktury nebo objekty vytvořené běžícím programem a dočasně uložené v hlavní paměti převedeny na posloupnost bitů, kterou lze ukládat na perzistentní datové úložiště, přenášet po síti nebo i předávat mezi spuštěnými procesy. Opačný proces, při kterém jsou z posloupnosti bitů zpětně rekonstruována původní data se nazývá deserialiazace. [11]

Nástroje pro serialiaci objektů jsou součástí mnoha programovacích jazyků (PHP, Java, .NET a další), což však neplatí pro jazyk C++. Pokud bychom chtěli serializovat i v C++, jsme nuceni použít některou z dostupných knihoven, které nám tuto funkcionalitu dodatečně zajistí.

1.2 s11n

s11n je multiplatformní open-source knihovna zajišťující serializaci objektů v jazyce C++. Autorem je knihovny Stephan Beal a v současné době je dostupná již její druhá stabilní verze pod označením 1.2.x.

1.2.1 Základní vlastnosti

- Snadno použitelné a do uživatelského kódu snadno zabudovatelné rozhraní pro serializaci široké škály datových typů jazyka C++.
- Podpora STL kontejnerů
- Možnost rozšiřovat knihovnu o vlastní datové typy
- Knihovna podporuje několik typů výstupních formátů – textový soubor, binární soubor a několik dialektů formátu XML. Výčet podporovaných formátů může být dále rozšířen pomocí přídavných balíčků.
- Jádro knihovny pracuje nezávisle na zvoleném výstupním formátu. O vstupně-výstupní operace se stará I/O vrstva, která od jádra oddělena a může být nahrazena i jiným, než standardně používaným řešením.

- Knihovna je psána dle standardních norem jazyka C++, čímž je zajištěna její snadná přenositelnost. Pro překlad knihovny musí být použit překladač s podporou šablon.
- Na platformě Unix je možné na výstupní data použít kompresní formáty zlib nebo b2lib. [7]

1.2.2 Serializace tříd

Vytvoření serializovatelné rřídy se skládá ze dvou kroků:

- Vytvoření operátorů pro serializaci a deserializaci:

```
class SerializovatelnaTrida{
// operator pro serializaci:
virtual bool operator()(s11nlite::node_type & cil) const;
// operator pro deserializaci:
virtual bool operator()( const s11nlite::node_type &
zdroj);
// ... };
```

- Registrace vytvořené třídy. Registrace se provádí pomocí tzv. supermaker ve tvaru:

```
#define S11N_TYPE ASerType
#define S11N_TYPE_NAME "ASerType"
#include <s11n.net/s11n/reg_s11n_traits.hpp>
```

Nyní může být objekt uložen pomocí funkce:

```
s11nlite::save( SerializovatelnaTrida, ''soubor.s11n'' );
```

1.3 Boost Serialization

Knihovna je součástí balíku knihoven Boost C++ Libraries. Jejím autorem je Robert Ramey. V současnosti je k dispozici verze 1.42.

1.3.1 Základní vlastnosti

- Přenositelnost kódu – knihovna je psána dle normy ANSI C++.
- Úspornost kódu – využívá standardních možností jazyka C++ jako je RTTI, šablony a vícenásobná dědičnost.
- Nezávislé označení pro každou verzi třídy, což zajistí, že pokud dojde ke změně definice třídy, lze do ní snadno importovat i data určená pro původní verzi třídy.
- Ukládání a načítání ukazatelů zároveň ukládá i načítá data, na které ukazují.

- Podpora serializace STL kontejnerů a ostatních běžně používaných šablon.
- Datová přenositelnost – datové proudy vytvořené na jedné platformě jsou vždy zpracovatelné i na ostatních platformách.
- Třídy, které mají být serializovány, nemusí být odvozeny od určité základní třídy ani do nich nemusí být implementovány žádné dodatečné metody. [5]

1.3.2 Výstupní formáty

Knihovna umožňuje ukládat data v textových souborech, v souborech formátu XML a v binárních souborech. Tyto výstupní formáty jsou v terminologii knihovny označeny jako archivy. Pro každý druh archivu jsou v knihovně definovány dvě třídy – jedna pro ukládání dat do souborů a druhá pro jejich opětovné načítání. Všechny tyto předdefinované třídy jsou připraveny k použití bez dalších nutných zásahů, v případě potřeby však mohou být dále rozšířeny a sloužit jako základ pro nové, uživatelem implementované výstupní formáty.

Pokud bude chtít programátor ve svém projektu ukládat data do některé z podporovaných formátů, musí k zdrojovému textu připojit příslušnou knihovnu odpovídající zvolenému formátu:

Textový soubor:
```
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
```

Textový soubor se širokými znaky:
```
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
```

XML soubor:
```
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
```

XML soubor se širokými znaky:
```
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
```

Binární soubor:
```
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
```

Prvně uvedená knihovna je určená pro ukládání dat do souboru, druhá pro jejich načítání ze souboru.

Instance jednotlivých tříd jsou vytvářeny pomocí přetíženého konstruktoru, kterému předáme vhodný vstupní/výstupní proud. Instanci třídy pro ukládání/načítání do textového souboru můžeme vytvořit pomocí následujícího kódu:

```
// pro ukládání
std::ofstream vyst_pr("nazev_souboru.txt");
boost::archive::text_oarchive vyst_ar(vyst_pr);

// pro načítání
std::ifstream vst_pr("nazev_souboru.txt");
boost::archive::text_iarchive vst_ar(vst_pr);
```

Obdobným způsobem lze vytvářet instance tříd i pro jiné druhy archivů.

1.3.3 Serializace tříd

Pro zápis nebo načtení tříd do archivu lze použít dvojici operátorů ‚<<' a ‚>>', které pomocí šablony vygenerují kód pro zpracování jednotlivých datových členů třídy. Tato šablona musí být programátorem vytvořena pro každou třídu, která má být serializována.

Šablona může být vypadat například takto:

```
template<class Archive>
void serialize(Archive & ar, const unsigned int version)
{
    ar & m_data1;
    ar & m_data2;
    ar & m_data3;
}
```

Operátor & bude v závislosti na tom, zda-li je archiv `ar` vstupní nebo výstupní, nahrazen příslušným operátorem pro zápis nebo načítání jednotlivých členů třídy `m_data1…3`.

Třídy pak budou ukládány/načítány do/z archivu pomocí následujícího kódu:

```
vystupni_archiv << instance_tridy;
vstupni_archiv >> instance_tridy;
```

1.4 Eternity

Eternity je malá knihovna, skládá se z pěti tříd ve třech zdrojových souborech. Při vývoji byl hlavní důraz kladen na multiplatformnost, používání výhradně standardních C++ knihoven a podporu výstupu v binární podobě a ve formátu XML. Knihovna využívá možností jazyka C++, jako jsou šablony nebo systém RTTI.

Vývoj Eternity v posledních letech nepokračuje. Aktuálně dostupná verze s označením 4.0 byla uvedena už v roce 2003.

1.4.1 Základní vlastnosti

- Podpora serializace objektů do binárních souborů nebo do formátu XML.
- Využívá pouze knihovny standardně dostupné v jazyce C++.
- Umožňuje dynamicky vytvářet a perzistentně ukládat všechny třídy bez jakýchkoliv omezení, včetně tříd s vícenásobnou dědičností a šablon.
- Třídy nemusí být odvozeny od nadřazené základní třídy
- Objekty, na které ukazuje více ukazatelů budou serializovány pouze jednou. [6]

1.4.2 Dynamická tvorba tříd

V knihovně je implementován vlastní systém pro dynamické vytváření instancí tříd. Základ systému tvoří sada šablon třídy `factory<class T>.` Tyto šablony mají dvě položky: instanci třídy T, kterou je potřeba dynamicky vytvořit a ukazatel na vrchní prvek seznamu čítajícího všechny instance `factory`<>, které se aktuálně nacházejí v paměti.

Třída `factory`<> má implementovány následující metody:

- Konstruktor třídy, který na vrchol seznamu instancí vloží ukazatel na aktuální instanci. Destruktor třídy pak provádí operaci opačnou.
- Metodu `create()`, která má jako vstupní parametr název třídy. Tento název je pomocí systému RTTI porovnáván s názvem instance třídy T. Pokud dojde ke shodě názvů, pak metoda vytvoří instanci třídy T a vrátí ukazatel na ni.

Použití systému ilustruje následující příklad:

```
factory<NovyObjekt> AFactory;
NovyObjekt *objekt = NULL;
string NazevTridy = “NovyObjekt”;
create(NazevTridy, &object);
```

Samotné vytvoření provádí funkce create, jejímiž parametry jsou uživatelem definovaný název třídy a ukazatel na prázdný objekt. [6]

1.4.3 Binární serializace

Pro podporu binární serializace je nutné do serializovatelné třídy implementovat metodu serialize(). Implementace této metody ukazuje následující příklad:

```
void AClass::serialize( bin_archive &stream)
{
  if (stream.is_loading())
  {
    stream >> m_nField1;
    stream >> m_fField2;
    stream.get_object(m_pField3);
  }
  else
  {
    stream << m_nField1;
    stream << m_fField2;
    stream.put_object(m_pField3);
  }
}
```

Parametr `stream` značí referenci na instanci třídy, která je odvozena od třídy `bin_archive` a která představuje archiv, do kterého ukládáme data. Jednotlivé členy třídy lze do archivu přidávat nebo je z něj načítat pomocí dvojice operátorů – ‚>>' pro ukládání a ‚<<' pro načítání. Tyto operátory však nelze používat u ukazatelů. Pokud chceme archivovat ukazatele, jsme nuceni použít funkci `put_object()` pro jejich uložení nebo funkci `get_object()` pro jejich načtení z archivu.

Nyní, pokud budeme chtít objekt serializovat stačí jen vytvořit instanci třídy `file_archive`, otevřít ji pro zápis, a poté ji předat jako parametr do vytvořené funkce `serialize()`. Popsaný postup znázorňuje následující ukázkový kód:

```
file_archive fa;
fa.open("file.bin", archive::store);
myObject.serialize(fa);
fa.close();
```

Výše uvedený kód uloží objekt do souboru zadaného prvním parametrem funkce `fa.open()`. Pokud budeme chtít objekt ze zadaného souboru načíst, jako druhý parametr funkce zadáme `archive::load`. [6]

1.4.4 Serializace do souborů formátu XML

Zajištění podpory serializace do XML souborů podobně jako u binárních souborů opět spočívá v implementaci metody `xml_serialiaze()`:

```
void AClass::xml_serialize( xml_archive &xml)
{
  if (xml.is_loading())
  {
```

```
    xml.read("Field1", m_nField1 ,0);
    xml.read("Field2", m_fField2, 0);
    xml.get_object("Field3", m_pField3 ,0);
  }
  else
  {
    xml.write("Field1", m_nField1);
    xml.write("Field2", m_nField2);
    xml.put_object("Field3", m_pField3);
  }
}
```

Parametr `xml` značí referenci na aktuálně používanou instanci třídy `xml_archive`, do které se budou data ukládat. Pro ukládání a načítání je nutné použít funkce `write()` a `read()`. Parametry těchto funkcí je člen třídy, který chceme uložit (nebo načíst) a jeho identifikační popisek, pod kterým bude uložen ve výstupním XML souboru. Pro ukládání a načítání ukazatelů na objekt lze opět použít funkce `get_object()` a `put_object()`. Parametry těchto funkcí jsou stejné jako u funkcí `write()` a `read()`.

Serializace objektu se provádí téměř identicky jako v případě binární serializace, pouze místo instance třídy `fa_archive` vytvoříme instanci třídy `xml_archive`. [6]

1.5 wxXmlSerializer

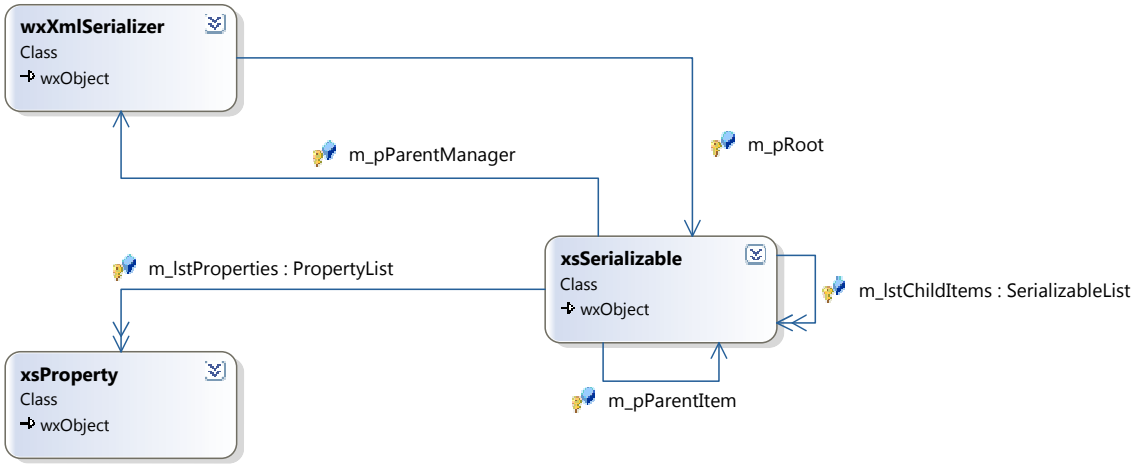
wxXmlSerializer je multiplatformní open-source knihovna postavená na knihovně wxWidgets. Knihovna umožňuje vytvářet perzistentní hierarchické datové kontejnery vhodné pro ukládání instancí tříd. Třídy mohou být hierarchicky upořádány a ukládány do XML struktury, která může být uložena na disku nebo v kterémkoli jiném výstupním proudu. [2]

1.5.1 Základní vlastnosti knihovny

Knihovna wxWmlSerializer se skládá ze třídy základních tříd:

- **wxXmlSerializer** – hlavní správce objektů, metody této třídy provádějí serializaci a deserializaci datových objektů.
- **xsSerializable** je základní třída pro všechny serializovatelné třídy, tedy ty třídy, které mají být serializovány. Třída poskytuje funkcionality nezbytné pro hierarchické uspořádání instancí, vstupně-výstupní operace a také uchovává informace o serializovaných členech jednotlivých serializovatelných instancí.

- **xsProperty** zapouzdřuje jednotlivé datové členy serializovatelných tříd. Ukládá informace o paměťových adresách datových členů, jejich datové typy, výchozí hodnotě, jejich název ve výstupním XML souboru a značku informující jestli má být člen serializován nebo ne. [2]



Obr. 1. Vztahy mezi jednotlivými třídami

Knihovna podporuje následující datové typy:

- Obecné datové typy jako jsou int, long, float, double, char, bool
- Nejčastěji používané datové typy z knihovny wxWidgets - wxString, wxPoint, wxSize, wxRealPoint, wxPen, wxBrush, wxFont, wxColour
- wxArrayString, pole proměnných datového typu wxRealPoint a pole proměnných obecných datových typů
- Dynamické a statické instance serializovatelných tříd [2]

1.5.2 Vytváření serializovatelných tříd

Všechny třídy, které mají serializovány, musí být odvozeny od základní třídy xsSerializable. Při implementaci třídy pak uživatel může pomocí sady maker označit ty členy, které budou serializovány. Tyto makra se postarají o vytvoření instance třídy xsProperty pro označený člen třídy.

Pro tento účel je možné použít makra ve tvaru:

- Univerzální makra:
  ```
  XS_SERIALIZE(clen_tridy, polozka)
  XS_SERIALIZE_EX(clen_tridy, polozka, vychozi)
  ```

- Makra pro specifický datový typ:
  `XS_SERIALIZE_INT(clen_tridy, polozka)`
  `XS_SERIALIZE_INT_EX(clen_tridy, polozka, vychozi)` [2]

Parametr `polozka` umožňuje definovat identifikační název, pod kterým bude datový člen uložen ve výstupní XML struktuře. Makra s příponou `_EX` umožňují zvolit výchozí hodnotu datového členu, ten pak bude serializován pouze tehdy, pokud se jeho aktuální hodnota liší od výchozí.

Příklad implementace serializovatelné třídy:

```
class SerializovatelnaTrida : public xsSerializable
{
     int m_iDatovyClen = 100;
     XS_SERIALIZE_INT(m_iDatovyClen, wxT(“data_int”);
};
```

1.5.3 Serializace objektů

Nejdříve je nutné vytvořit instanci třídy `wxXmlSerializer`, která bude sloužit jako serializer, do kterého budou vkládány objekty určené k serializaci.

Objekty je možné vkládat do serializeru několika způsoby:

- Objekt bude připojen jako kořenový uzel serializeru – vhodné pro samostané objekty. Objekt lze připojit pomocí funkce `wxXmlSerializer::SetRootItem()`, kde parametrem funkce je ukazatel na serializovaný objekt.
- Objekty budou připojeny ke kořenovému uzlu serializeru – seznam objektů
- Objekty budou připojeny k jakémukoliv již dříve připojenému objektu – stromová struktura objektů.

U posledně dvou jmenovaných případů je možné objekty vkládat do serializeru pomocí přetíženého operátotu ‘<<’ nebo pomocí funkce `wxXmlSerializer::AddChild()`.

Obsah serializer může být uložen do výstupního XML souboru pomocí funkce `wxXmlSerializer::SerializeToXml(const wxString &file)`. Parametrem funkce je název výstupního souboru.

Z uloženého XML souboru mohou být objekty načteny zpět pomocí funkce `wxXmlSerializer::DeserializeFromXml(const wxString &file)`, kde parametrem je cesta k uloženému XML souboru.

2 WXWIDGETS

2.1 Co je wxWidgets

wxWidgets je balík softwarových knihoven určených pro programovací jazyk C++, které umožňují vývoj aplikací pro několik různých softwarových platforem (Windows, MacOS, FreeBSD, GTK+). Pro každou z těchto platforem je k dispozici speciálně verze knihoven.

Kromě základních API funkcí pro tvorbu grafického rozhraní aplikace poskytuje wxWidgets další sadu prvků vhodných pro přístup k dostupných funkcionalitám cílového operačního systému (práce se souborovým systémem, řízení procesů, atd.). Součástí knihovny jsou i třídy zjednodušující vývoj programů a implementující běžně používané algoritmy a technologie (síťové operace, práce s grafikou, podpora zobrazení HTML, atd. …).

Knihovna wxWidgets je šířena jako open-source projekt, je tedy komukoliv zdarma k dispozici pro nekomerční i komerční použití. [1]

2.2 Historie wxWidgets

Počátek vývoje knihovny wxWidgets se datuje do roku 1992 na Institutu aplikací umělé inteligence edinburské univerzity. Jeden ze studentů tohoto ústavu, Julian Smart, tou dobou vyvíjel projekt, u kterého byl vyžadován běh jak v prostředí MS Windows, tak i v Unixových X-Windows. Všechny tehdejší komerčně dostupné multiplatformní knihovny však byly neúměrně drahé, a tak Julian Smart musel vytvořit knihovnu vlastní. Výsledkem jeho snažení byla knihovna wxWindows s podporou technologií XView a MFC 1.0 (knihovna MFC, pro jejíž používání je nutné vlastnit vývojové prostředí MS Visual Studio, však byl ve verzi pro Windows záhy nahrazen nativním rozhraním Win32API). Zdrojový text knihovny byl po svolení institutu umístěn k volnému stažení na internet, což zajistilo její brzký masový rozvoj. [1]

2.3 Vlastnosti wxWidgets

2.3.1 Základní třídy pro tvorbu GUI

wxWidgets obsahují velké množství tříd určených pro tvorbu standardních oken a ovládacích prvků. Většina těchto tříd zapouzdřuje standardně dostupné ovládací prvky daného operační systému, některé třídy však implementují nové, v systému jinak

nedostupné prvky, jako např. mřížka (wxGrid) nebo rozšířený textový ovládací prvek (wxStyledCtrl). [1]

2.3.2 Zpracování zpráv

Všechny třídy odvozené od třídy wxEvtHandler mohou reagovat na veškeré události generované operačním systémem nebo aplikacemi. Navíc lze využívat i dědičnost – události zpracovávané jednotlivými ovládacími prvky jsou předávány jejich rodičovským prvkům.

Mapování zpráv mezi událostmi a příslušnými obslužnými funkcemi lze provést dynamicky (funkce `Connect()`) nebo staticky pomocí tzv. mapy zpráv. [1]

2.3.3 Kontexty grafických zařízení

Knihovna wxWidgets, podobně jako OS Windows, využívá ke kreslení na plochu okna aplikace kontexty zařízení. Třída wxDC zapouzdřuje kontext pro kreslení na plochu okna, wxPrinterDC poskytuje výstup na tiskárnu, třída wxMemoryDC umožňuje kreslení do bitmap a tak dále. [1]

2.3.4 Tisk

Podpora pro tisk je ve knihovně wxWidgets zajištěna prostřednictvím tříd wxPrinter, wxPrintout a wxPreview. [1]

2.3.5 Bitmapy

Třída `wxBitmap` podporuje různé typy bitmapů v závislosti na jejich podpoře cílovou platformou. Třída `wxIcon` načítá ikony ve formátech ICO a XPM, třída `wxImage` podporuje formáty JPG, TIFF, PCX, PNG a BMP, načtené obrázky navíc dokáže rotovat a měnit jejich měřítko. [1]

2.3.6 Dialogy

Knihovna wxWidgets nabízí několik možností jak nadefinovat velikost, vlastnosti a rozmístění uživatelských prvků dialogového okna. Tato definice může být uložena přímo v programu nebo může být umístěna zvlášť v souboru XRC (formát XML).

K dispozici je i celá řada standardních systémových dialogů, jakými jsou wxFileDialog, wxDirDialog, wxMessageBox, wxColourDialog, wxFontDialog, wxPrintDialog a další. [1]

2.3.7 Datové struktury

wxWidgets obsahují některé běžně používané datové struktury, jakými jsou např. wxStringList, wxString a wxHashTable. Navíc nabízí alternativu k standardní knihovně jazyka C++, která není na všech platformách dostatečně implementována. [1]

2.3.8 Podpora ladění

wxWidgets nabízí skupinu speciálních funkcí použitelných pro zápis ladících informací a chybových stavů na standardní výstup, do debug streamu, nebo do souboru. Paměťové operace jsou tvořeny tak, že umožňují detekci případných úniků paměti. Tyto úniky pak mohou být vypsány po ukončení aplikace. [1]

2.3.9 Komunikace mezi procesy

Knihovna wxWidgets podporuje klasické socketové funkce i některé nejčastěji používané komunikační protokoly, jakou jsou HTTP nebo FTP. Na platformě Windows navíc knihovna podporuje protokol DDE. [1]

2.3.10 Databáze

wxWidgets podporuje celou řadu databází prostřednictvím svých tříd využívajíchc standard ODBC. Navíc lze využít databázi Xbase, což je klon standardní dBase. [1]

2.4 Instalace knihovny wxWidgets

2.4.1 Distribuce knihovny

Knihovnu wxWidgets je možné volně stáhnout z internetových stránek www.wxwidgets.org a to buď v podobě již předkompilovaných knihoven a hlavičkových souborů nebo ve formě zdrojových souborů knihovny, které je nutno před použitím nejdříve zkompilovat. [1]

2.4.2 Vývojové nástroje použitelné s wxWidgets

Projekty využívající knohoven wxWidgets lze kompilovat v následujících překladačích: VC++, Borland C++, Watcom C++, CodeWarrior, Cygwin, MinGW32, GCC.

Knihovna wxWidgets je podporována v celé řadě komerčních i freewarových vývojových prostředích.

Mezi nejlepší komerční vývojová prostředí patří MS Visual Studio, které umožňuje integrovat knihovnu wxWidgets včetně její nápovědy a ApplicationWizard pro automatizovanou tvorbu kostry aplikace.

Kvalitní freewarová vývojová prostředí jsou MinGW Developer Studio, wxDev-Cpp a Code::Blocks. Všechna tato prostředí jsou založena na překladači MinGW32, které mají integrován debugger a editor formulářů. [1]

II. PRAKTICKÁ ČÁST

3 INSTALACE KNIHOVNY WXXMLSERIALIZER

3.1 Instalace na platformě MS Windows

Po stažení a rozbalení archivu s knihovnou lze v adresáři build spustit dávkový soubor `create_build_files.bat`, který vytvoří projektové soubory pro vývojová prostředí CodeLite, Code::Blocks a MS Visual Studio. Tyto automaticky vytvořené soubory obsahují všechny vzorové projekty dodávané s knihovnou.

3.2 Instalace na platformě Unix

Instalace je zde podobná jako na platformě MS Windows. V adresáři build stačí spustit soubor `create_build_files.sh`, který vytvoří projektové soubory pro vývojová prostředí CodeLite a Code::Blocks.

4 TVORBA DOKUMENTACE KE KNIHOVNĚ

4.1 Popis API knihovny

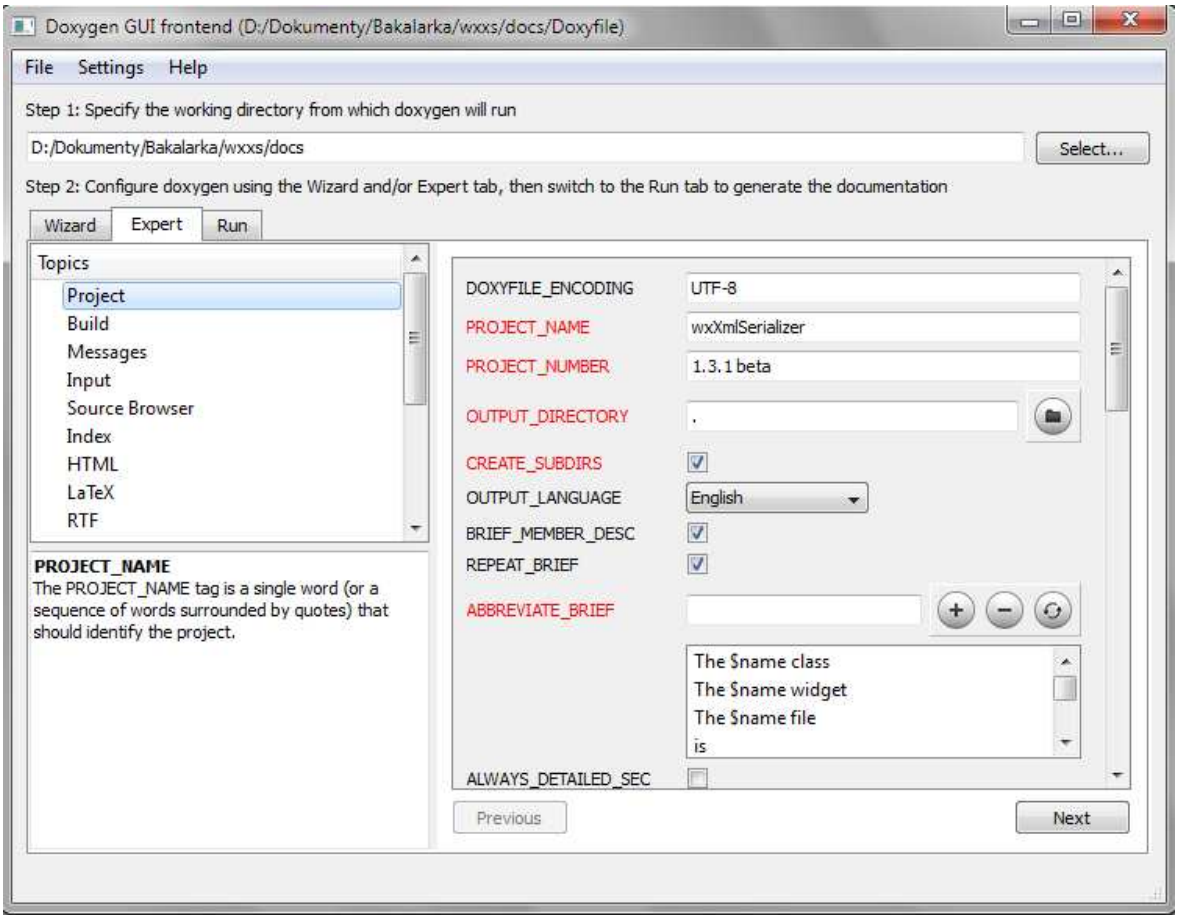
4.1.1 Dokumentační systém Doxygen

Dokumentace k API knihovny byla vytvořena pomocí dokumentačního systému Doxygen. Tento systém umožňuje z patřičným způsobem okomentovaných zdrojových souborů vytvářet projektovou dokumentaci v libovolném z podporovaných formátů, mezi které mimo jiné patří HTML, Latex nebo PDF.

4.1.2 Konfigurace systému Doxygen

Doxygen je aplikace, která se spouští z příkazové řádky a její konfiguraci lze provést pomocí konfiguračního skriptu, který je Doxygenu předáván jako parametr při spouštění. Pomocí tohoto konfiguračního skriptu lze nastavit cestu ke zdrojovým souborům, druh výstupního formát, zda-li budou v dokumentaci vykresleny a podobně. Veškeré konfigurační parametry jsou podrobně popsány v dodávané dokumentaci.

Konfigurační skript je možné vytvářit v běžném textovém editoru nebo lze jít podstatně pohodlnější cestou a použít dodávaný nástroj Doxywizard, který zároveň slouží jako GUI rozhranní pro z příkazové řádky spouštěný Doxygen. V Doxywizardu jsou jednotlivé konfigurační parametry členěny do logických sekcí, po najetí kurzoru na parametr se navíc zobrazí i nápověda. Po nastavení všech parametrů a uložení konfiguračního skriptu lze přímo z Doxywizardu spustit Doxygen a to pomocí tlačítka „Run doxygen“ v záložce Run.



Obr. 2. Konfigurační utilita Doxywizard

4.1.3 Komentář ve zdrojových kódech

Doxygen generuje dokumentaci z komentářů umístěných přímo ve zdrojových kódech. Tyto komentáře musí být psány v předepsaném formátu. Pro zdrojový kód jazyka C++ může komentář vypadat následovně:

```
/*!
 * …komentář…
 */
```

Do komentáře mohou být vkládány i speciální příkazy, např. pro specifikaci druhu vloženého komentáře, vložení obrázku, vložení zdrojové kódu, atd. Podrobný popis všech speciálních příkazů je uveden v dokumentaci k Doxygenu.

4.1.4 Struktura vygenerovaných HTML stránek

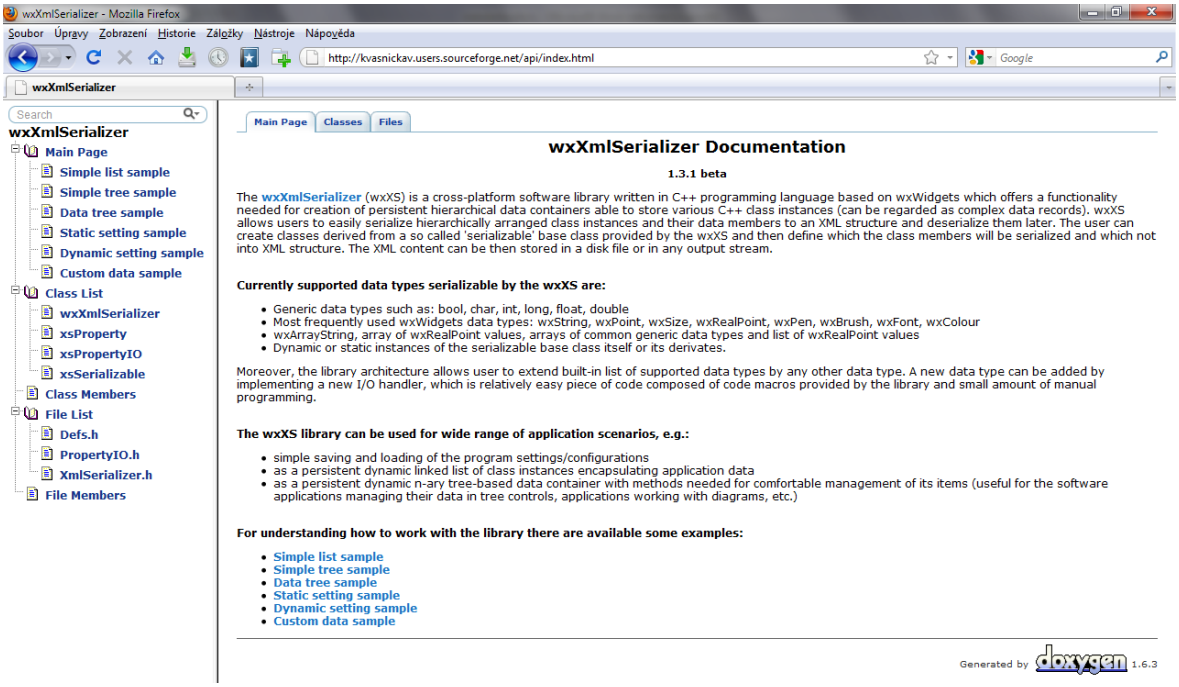
Stránka je rozdělena do dvou oddílů. Levá část obsahuje navigaci a kolonku pro vyhledávání na stránkách. Navigace byla na stránky přidána nastavením

konfiguračního parametru `GENERATE_TREEVIEW` a vyhledávání parametrem `SEARCHENGINE`. Obě přidané komponenty ke své funkci používají Javascript.

V hlavním framu se zobrazuje obsah stránek samotných.

4.1.5 Úvodní stránka dokumentace

Úvodní stránka dokumentace API je vygenerována z textové souboru mainpage.txt. Hned na začátku souboru je napsán příkaz `/mainpage`, který značí, že celý blok komentáře, ve kterém byl příkaz použit, bude vypsán na úvodní stránce dokumentace (index.html).



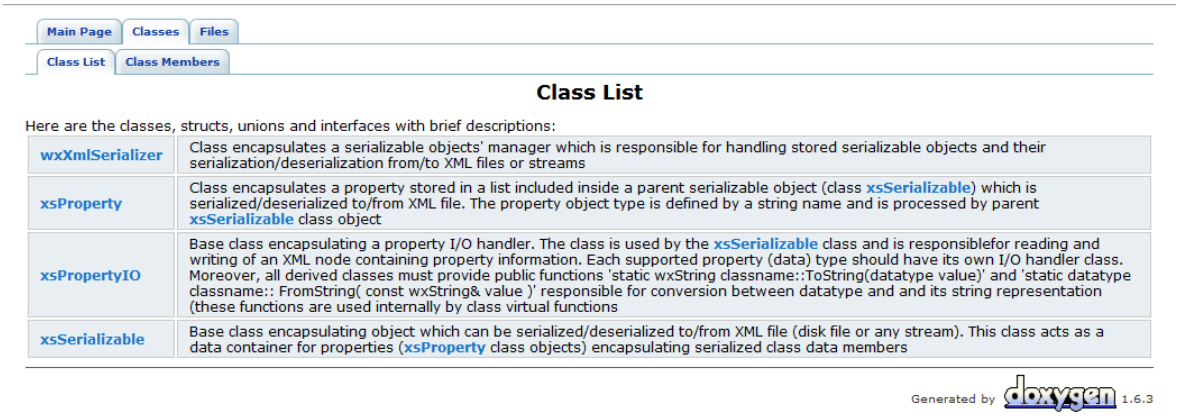
Obr. 3. Úvodní stránka dokumentace API

V závěru úvodní stránky jsou uvedeny odkazy na popis vzorových příkladů použití knihovny. Odkazy byly pomocí příkazu `\subpage` ve tvaru `\subpage simple-list-sample Simple list sample`, kde parametr druhý v pořadí určuje název stránky, na který odkaz odkazuje. Jednotlivé stránky s popisem příkladů jsou exportovány ze samostatných textových souborů. Na začátku každého z těchto souboru je použit příkaz `\page` ve stejném tvaru jako `\subpage` na hlavní stránce. Tento příkaz slouží ke spojení stránky s příslušným odkazem na hlavní stránce.

4.1.6 Popis tříd

Pro popis tříd je v kódu použit příkaz `/brief`, který určuje, že blok komentáře, ve kterém byl příkaz použit, se vztahuje k deklaraci třídy následující hned po tomto bloku.

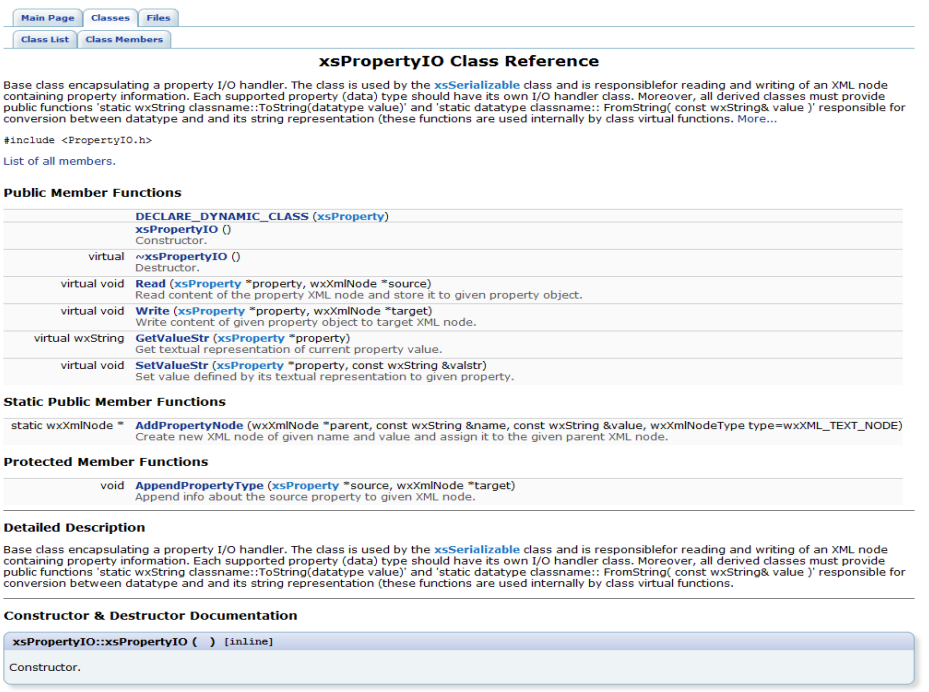
Komentář ke třídě je ve výsledné dokumentaci uveden v seznamu tříd a v podrobném popisu dané třídy.



Obr. 4. Seznam tříd

4.1.7 Popis členských funkcí a proměnných

Podrobný popis třídy je na stránkách přístupný po kliknutí na příslušnou funkci v seznamu tříd. Popis obsahuje seznam všech členů a jejich podrobný popis.



Obr. 5. Stránka s podrobným popisem třídy xsPropertyIO

V komentářích obsahujících popis členů třídy jsou kromě příkazu `/brief` použity ještě příkazy:

- `/param` – popis vstupních parametrů funkce nebo třídy
- `/return` – popis návratových hodnot funkce

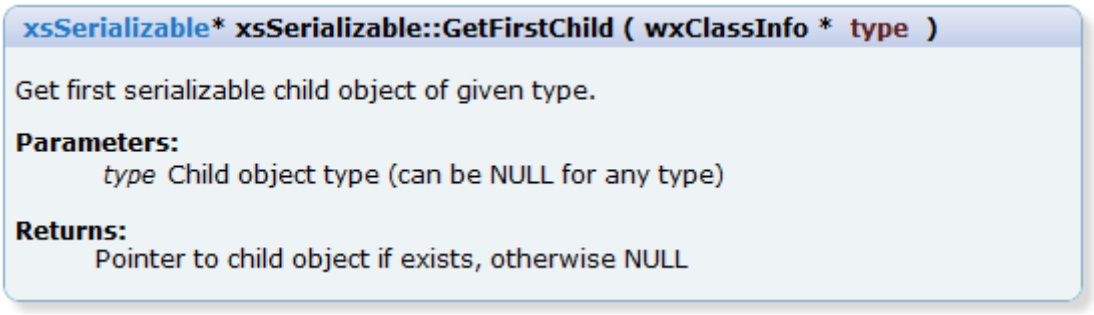
Komentář k funkci pak ve zdrojovém kódu vypadá například následovně:

```
/*!
 * \brief Get first serializable child object of given type.
 * \param type Child object type (can be NULL for any type)
 * \return Pointer to child object if exists, otherwise NULL
 */
xsSerializable* GetFirstChild(wxClassInfo *type);
```

Z výše uvedeného komentáře systém Doxygen vytvoří položku do seznamu členů třídy. Tato položka obsahuje pouze název funkce a její popis, uvedený za příkazem `\brief`.

Obr. 6. Položka v seznamu členů třídy

Dále je z komentáře vytvořena podrobnější tabulka, která již kromě názvu a popisu funkce obsahuje i popis vstupních parametrů a návratových hodnot:

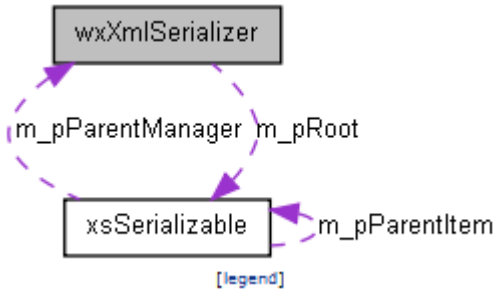


Obr. 7. Tabulka s popisem členské funkce

Pokud je v konfiguračním souboru nastaven parametr `EXTRACT_ALL` na hodnotu `YES`, bude výše uvedená tabulka a položka v seznamu vytvořena i pro ty třídy a funkce, které nejsou okomentovány. Tabulka i položka budou pochopitelně pouze s názvem funkce bez dalšího popisu.

4.1.8 Generování diagramů

Systém Doxygen poskytuje možnost automatického generování digramů znázorňujících vztahy mezi jednotlivými třídami v projektu. Diagramy jsou vytvářeny pomocí interního generátoru nebo lze použít některých z externích nástrojů pro tvorbu grafů. Diagramy, které jsou součástí dokumentace API knihovny wxXmlSerializer byly vytvořeny pomocí externího nástroje Graphviz. O obsluhu nástroje se stará samotný Doxygen, kterému stačí do konfiguračního souboru zadat správnou cestu ke spouštěcímu souboru Graphvizu. Diagramy jsou v dokumentaci vyobrazeny vždy na začátku stránky s podrobným popisem třídy. Zapnutí podpory generování diagramů a další příslušné parametry lze pomocí utility Doxywizard nastavit v položce Expert -> Dot.

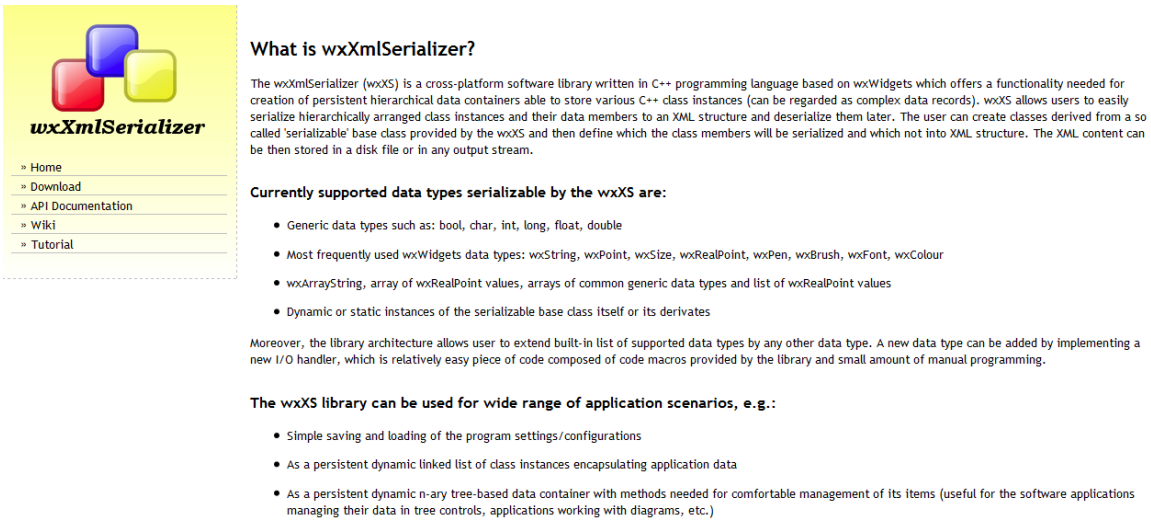


Obr. 8. Diagram znázorňující vztah mezi třídami wxXmlSerializer a xsSerializable

4.2 Webové stránky knihovny wxXmlSerializer

4.2.1 Základní rozvržení stránek

Stránka je rozdělena do dvou částí – navigace pro pohyb na stránkách a část, do které je vypisován textový obsah stránek. Jednotlivé komponenty stránek (navigace, úvodní stránka, stránky tutoriálu) jsou uloženy v samostatných HTML souborech. Stránka jako celek je pak poskládána pomocí PHP funkce `include_once()`, která v místě, kde je zadaná, vypíše obsah HTML souboru předaného v parametru funkce.



Obr. 9. Úvodní stránka

4.2.2 Navigace

Navigace slouží pro rychlý pohyb po stránkách, obsahuje odkazy na úvodní stránku, odkaz na stažení knihovny ze Sourceforge, odkaz na dokumentaci k API knihovny, odkaz na Wiki stránky knihovny a rozevíratelný odkaz na jednotlivé stránky tutoriálu obsahující popis základní práce s knihovnou.

Součástí navigačního panelu je i logo ve formátu PNG s průhledným, pro jehož základ jsem použil logo wxWidgets, stažené ze stránek http://wx.ibaku.net/logo/. Toto logo je volně dostupné pod licencí Creative Commons Share Alike 1.0 Generic.

Obr. 10. Navigační panel

Navigační panel je umístěn v HTML souborech `navigation.html` a `navigation-open.html`. Druhý jmenovaný soubor obsahuje navigaci s otevřeným tutoriálem.

Navigace je vytvořena pomocí HTML elementu seznam, do kterého je vnořen další seznam s odkazy na jednotlivé stránky tutoriálu. Odkazy na tyto stránky obsahují jako PHP parametr názvy jednotlivých HTML souborů se stránkami tutoriálu bez koncovky html. Pomocí těchto parametrů pak jednoduchý PHP skript umístěný v souboru `userguide.php` zobrazí zvolenou stránku.

Použitý PHP skript vypadá následovně:

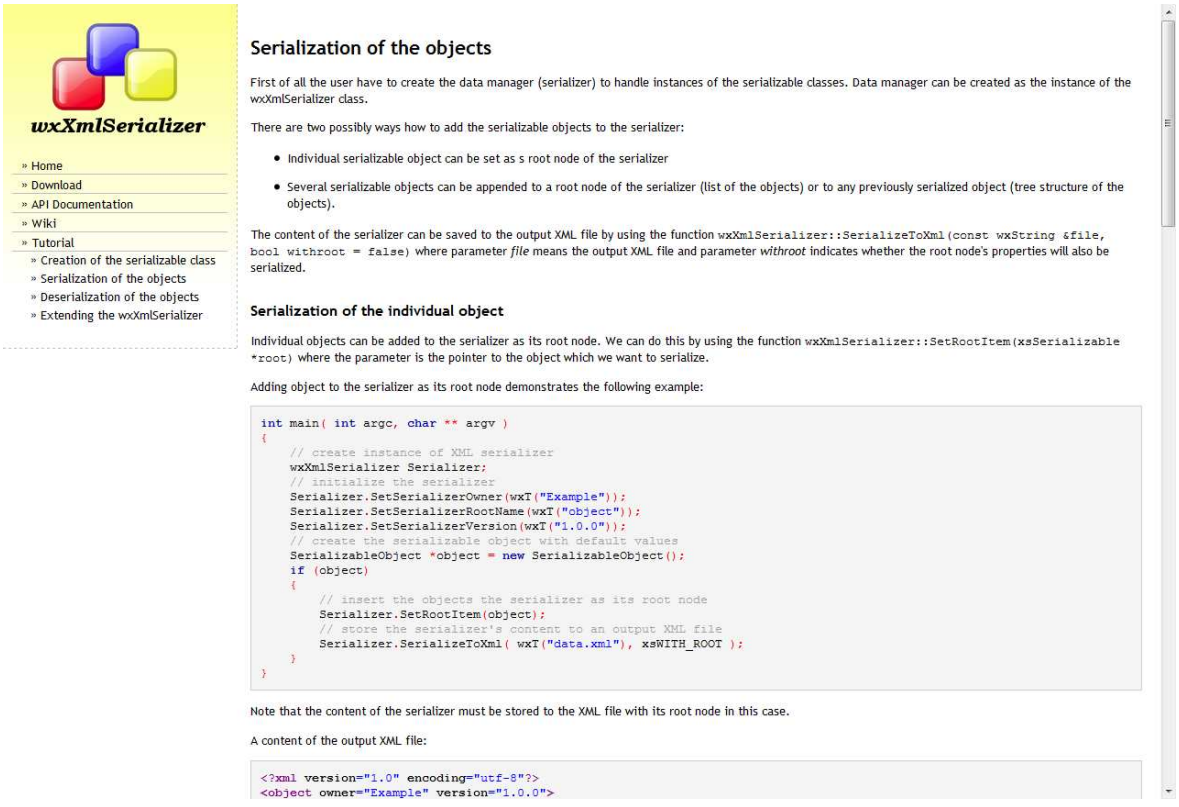
```
<?php
  $file="./_html/".$_GET['page'].".html";
  if (file_exists($file))
  {
    include_once($file);
  }
  else
  {
    include_once("./_html/serializableclass.html");
  }
?>
```

Z parametru předaného odkazem skript nejdříve vytvoří proměnnou typu string, do které doplní příponu html a název adresáře s HTML soubory. Poté skript ověří, zda-li soubor skutečně existuje. Pokud ano, je zakomponován do zobrazované stránky, pokud ne, bude do stránky vypsána první stránka tutoriálu (Creation of the serializable class).

4.2.3 Tutoriál

V tutoriálu je popsán základní způsob práce z knihovnou wxXmlSerializer. Je zde popsána tvorba serializovatelných tříd, serialiazace/deserializace objektů a tvorba nových serializovatelných datových typů.

Jak již bylo výše uvedeno, stránkami tutoriálu lze procházet přes odkazy v navigaci. V tutoriálu jsou uvedeny i ukázky zdrojových kódu a jimi vytvořených výstupních XML souborů. Tyto kódy byly vyexportovány do formátu HTML pomocí vývojového prostředí Code::Blocks a jeho modulu Source Exporter.



Obr. 11. Ukázka stránky z tutoriálu

4.2.4 Umístění stránek na Sourceforge

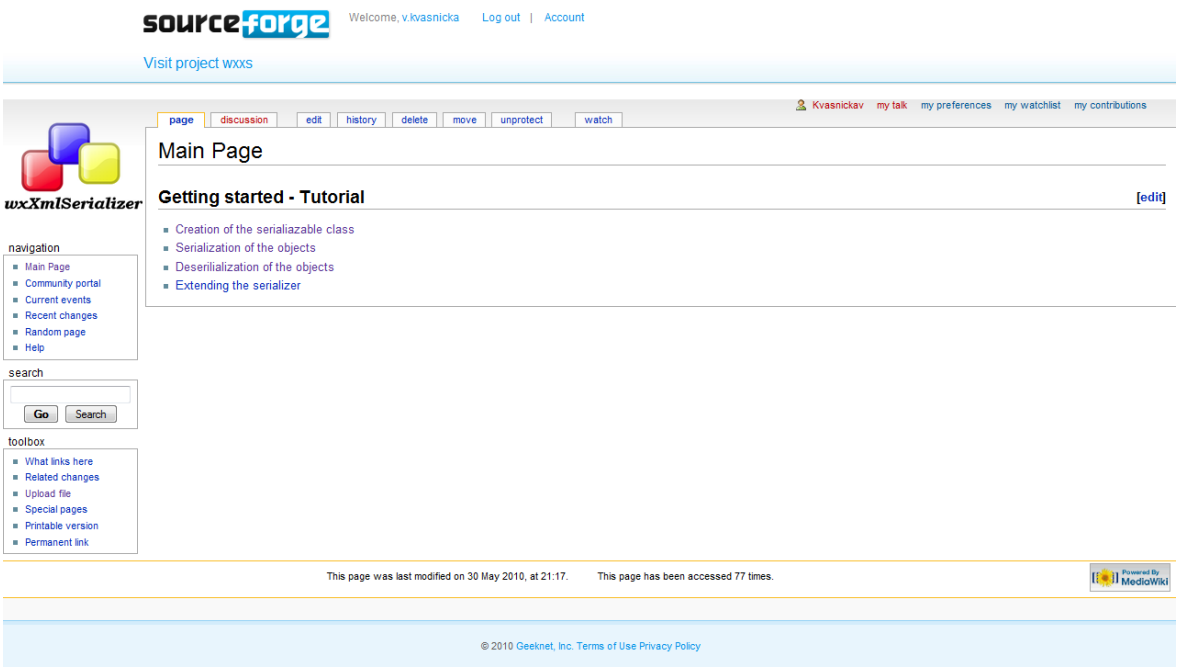
Sourceforge poskytuje uživatelům prostor pro webové stránky svých projektů. Webový server podporuje mimo jiné technologie i skriptovací programovací jazyk PHP, který je použit i na webových stránkách knihovny wxXmlSerializer.

Webové stránky byly na server nahrány přes SFTP protokol pomocí GUI klienta FileZilla. Přihlašovací údaje a umístění stránek na serveru je následující:

- Název hostitele: www.sourceforge.net
- Použitý protokol: SFTP
- Číslo portu: 22
- Přihlašovací jméno je ve tvaru: prihlasovaci_jmeno,wxxs
- Umístění stránek na serveru: /home/groups/x/xs/wxxs/htdocs

4.3 Portál Wiki pro wxXmlSerializer

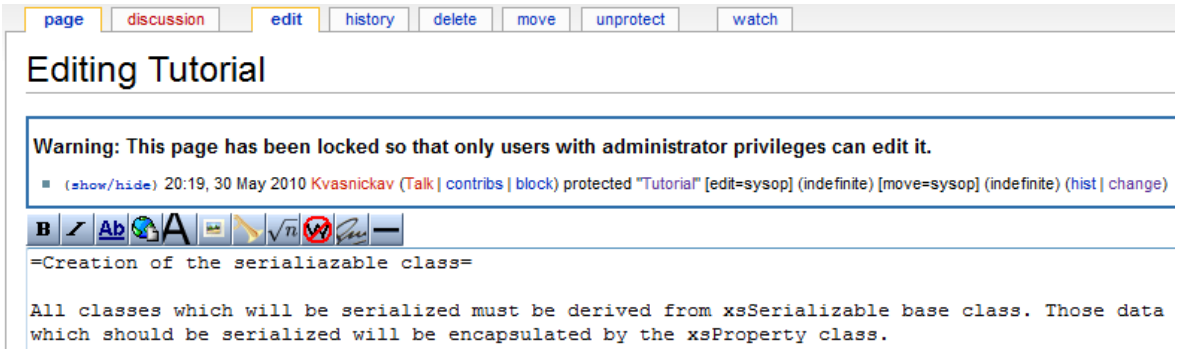
Wiki je přístupné po kliknutí na příslušný odkaz na webu knihovny. Wiki obsahuje tutoriál se základním popisem knihovny. Součástí tutoriálu je i diskuze, do které mohou přidávat příspěvky všichni návštěvníci stránek.



Obr. 12. Hlavní stránka portálu Wiki

4.3.1 Editace stránek na Wiki

Kliknutím na tlačítko edit se přejde do jednoduchého editoru, ve kterém lze upravovat obsah právě zobrazovaného článku. Formátování textu se provádí pomocí speciálních Wiki tagů, které lze do textu vkládat ručně nebo lze používat sadu editačních tlačítek, které budou tagy do textu vkládat automaticky.



Obr. 13. Editor článků na Wiki

Ukázka některých použitých Wiki tagů:

`==Nadpis druhé úrovně==`

`===Nadpis třetí úrovně===`

`'''tučný text'''`

`[[File:treesample.jpg]]` – vloží do článku obrázek umístěný na portálu Wiki

4.3.2 Vytváření nových článků

Existují dva způsoby jak vytvářet nové články na portálu Wiki:

- Do vyhledávače se napíše název nové stránky a na nalezené neexistující stránce se objeví výzva pro editaci nové stránky.
- Do existující stránky se vloží odkaz na novou stránku. Po kliknutí na tento odkaz se rovněž objeví výše uvedená stránka s výzvou pro editaci.

Pro vytvoření stránky s tutoriálem jsem použil druhou metodu. Na hlavní stránku jsem nejprve umístil odkazy na tutoriál přes které jsem se poté dostal k jeho editaci.

4.3.3 Vkládání obrázků

Obrázky je možné vkládat podobně jako nové stránky. Nejprve se vytvoří odkaz na neexistující obrázek, tento odkaz bude mít v textu červenou barvu. Po kliknutí na něj se objeví stránka, přes kterou je možné obrázek nahrát na server. Maximální velikost obrázku je 16 MB a podporovanými formáty jsou png, gif, jpg, jpeg a svg.

4.3.4 Nastavení uživatelských práv

Měnit obsah jednotlivých článku na portálu Wiki mohou jen ty skupiny uživatelů, kterým k tomuto úkonu bylo uděleno oprávnění. Nastavení oprávnění k editaci článků se nastavuje stisknutím tlačítka protect u zvoleného článku. Celkově je možné volit mezi třemi úrovněmi oprávnění: obsah článku mohou měnit všichni uživatelé, pouze registrování uživatelé nebo pouze uživatelé s administrátorskými právy.

Na portálu Wiki pro wxXmlSerializer mohou uživatelé bez omezení pouze přidávat příspěvky do diskuze u tutoriálu. U všech ostatních části mohou měnit obsah pouze uživatelé s administrátorskými právy.

ZÁVĚR

V teoretické části bakalářské práce jsem provedl srovnání čtyř multiplatformních open-source knihoven vhodných pro zajištění datové perzistence objektů jazyka C++. Při popisu jsem se zaměřil zejména na praktické použití knihoven, na náročnost implementace jednotlivých řešení do programátorských projektů.

Několik stránek teoretické části jsem věnoval i popisu wxWidgets, na kterých je postavena i knihovna wxXmlSerializer.

V praktické části jsem se nejdříve seznámil s možnostmi knihovny wxXmlSerializer. Získané znalosti jsem poté využil při tvorbě dokumentace ke knihovně.

Součástí praktické části byla též tvorba webových stránek, na kterých je umístěna veškerá dokumentace ke knihovně, která byla v rámce této bakalářské práce vytvořena. Webové stránky byly umístěny na SourceForge a jsou přístupné na adrese *http://wxxs.sourceforge.net.*

SEZNAM POUŽITÉ LITERATURY

[1] BLIŽŇÁK, Michal. Systémové programování. 1. Vydání. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005. 202 s. ISBN 80-7318-364-1

[2] BLIŽŇÁK, M., DULÍK, T., VAŠEK, V. A Persistent Cross-Platform Class Objects Container for C++ and wxWidgets. WSEAS TRANSACTION on COMPUTERS, Issue 5, Volume 8, May 2005. s. 778-787. ISSN 1109-2750.

[3] KOSEK, Jiří. HTML - tvorba dokonalých WWW stránek - podrobný průvodce. 1. vydání. Praha: Grada Publishing, 1998. 296 s. ISBN 80-7169-608-0.

[4] MCFARLAND, David Sawyer. CSS – chybějící manual. 1. Vydání. Praha: Grada Publishing, a. s., 2007. 432 s. ISBN 978-80-247-2122-4

[5] RAMEY, Robert. Boost Serialization [on-line]. 2009. [cit. 25. 5. 2010]. Dostupný z WWW:
<http://www.boost.org/doc/libs/1_43_0/libs/serialization/doc/index.html>

[6] SANTI, Nicola. Eternity Official guide 4.0. version [online]. 11. 4. 2003 [cit. 25. 5. 2010]. Dostupný z WWW: < http://sourceforge.net/projects/eternity-it/>

[7] BEAL, Stephan. s11n - an Object Serialization Framework for C++ [online]. 27. 4. 2008 [cit. 25. 5. 2010].
Dostupný z WWW: < http://www.s11n.net/download/#s11n>

[8] *Doxygen Manual* [online]. [cit. 25. 5. 2010].
Dostupný z WWW: < http://www.doxygen.nl/manual.html>

[9] *wxWidgets 2.8 Online Manual* [online]. [cit. 25. 5. 2010]
Dostupný z WWW: < http://docs.wxwidgets.org/stable/>

[10] GABHART, Kyle. J2EE pathfinder: Persistent data management, Part 1 [online] 2. 4. 2003 [cit. 25. 5. 2010]. Dostupný z WWW:
<http://www.ibm.com/developerworks/java/library/j-pj2ee3.html>

[11] *Serialization* [online]. 23. 5. 2010 [cit. 25. 5. 2010]. Dostupný z WWW:
<http://en.wikipedia.org/wiki/Serialization>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface.
FTP	File Transfer Protocol
GUI	Graphical user interface
HTML	HyperText Markup Language.
PDF	Portable document format
PHP	PHP: Hypertext Preprocessor
PNG	Portable Network Graphics
SFTP	SSH File Transfer Protocol
XML	Extensible Markup Language

SEZNAM OBRÁZKŮ