Kristýna-GIS

Veřejné rozhraní zásuvných modulů pro mapové služby

Specifikace rozhraní verze 1.0

Kristýna-GIS

Uvolněno: Říjen 2005
Revize: Březen 2021

Obsah

1. Úvod

Dokument popisuje rozhraní pro dynamicky připojované knihovny, které může být použito pro čtení a vykreslování map poskytovaných internetovými/intranetovými mapovými službami. Rozhraní Vám dovoluje vytvořit Vaší vlastní knihovnu a tímto způsobem můžete zajistit komunikaci mezi mapovou službou a Kristýnou. Pokud vytvoříte svojí vlastní knihovnu, doporučujeme vytvořit také instalační program, který knihovnu bude instalovat. V kapitole *Instalace Vaší vlastní knihovny* naleznete několik doporučení.
Toto je popis rozhraní verze 1.0. Rozhraní je podporováno Kristýnou-GIS verze 2.0 a vyšší. Kristýna-GIS prohlížečka toto rozhraní nepodporuje. Tento materiál popisuje informace dostupné v čase jeho publikování a je v podstatě totožný s informacemi, které jsou obsaženy v nápovědě Kristýny. V některých případech můžete najít aktuálnější informace na internetových stránkách Kristýny (www.christine-gis.com).

2. Použité datové typy

Většina použitých datových typů jsou běžné datové typy používané v Microsoft Windows API.

BOOL - Logický typ (nabývá hodnot TRUE nebo FALSE), je uchováván jako celočíselný typ se znaménkem

COLORREF - 32-bitová hodnota používaná pro specifikaci RGB barvy. Nejméně významný byte obsahuje hodnotu relativní intenzity červené složky, druhý byte obsahuje hodnotu pro zelenou, a třetí byte obsahuje hodnotu pro modrou. Nejvýznamnější byte musí být nulový. Maximální hodnota jednoho byte je 0xFF.

double - 64-bitové desetinné číslo se znaménkem

DWORD - 32-bitové celé číslo bez znaménka

EXTENT - Struktura, která definuje rozsah souřadnic X, Y, Z a M hodnot

```
typedef struct _EXTENT {
    double xMin;       // minimální hodnota souřadnice X
    double yMin;       // minimální hodnota souřadnice Y
    double xMax;       // maximální hodnota souřadnice X
    double yMax;       // maximální hodnota souřadnice Y
    double zMin;       // minimální hodnota souřadnice Z (ignorováno)
    double zMax;       // maximální hodnota souřadnice Z (ignorováno)
    double mMin;       // minimální hodnota hodnoty M (ignorováno)
    double mMax;       // maximální hodnota hodnoty M (ignorováno)
} EXTENT;
```

EXTENT* - Ukazatel na strukturu rozsahu

FALSE - Logická hodnota, uložena jako celé číslo se znaménkem s hodnotou nula

HWND - Popisovač okna

NULL - Nulová hodnota

POINT - Struktura definující x a y souřadnice bodu

```
typedef struct tagPOINT {
    LONG x;            // x souřadnice bodu
    LONG y;            // y souřadnice bodu
} POINT;
```

POINT* - Ukazatel na strukturu popisující souřadnice bodu

SIZE - Tato struktura specifikuje šířku a výšku obdélníku

```
typedef struct tagSIZE {
    LONG cx;     // šířka obdélníku
    LONG cy;     // výška obdélníku
} SIZE;
```

SIZE* - Ukazatel na strukturu popisující rozměry obdélníku

TCHAR* - Ukazatel na pole znaků (řetězec)

TRUE - Logická hodnota, uložena jako celé číslo se znaménkem s nenulovou hodnotou

void - Použito jako návratový typ funkce specifikuje, že funkce nevrací žádnou hodnotu

void* - Ukazatel na nespecifikovaný datový typ nebo strukturu ("univerzální ukazatel")

WORD - 16-bitové celé číslo bez znaménka

WMSCONNECTION - struktura uchovávající parametry spojení popřípadě veškerá jiná potřebná data

```
typedef struct _WMSCONNECTION {
    // povinná část struktury
    DWORD    dwSize;    // velikost této struktury (včetně nepovinné části)
    TCHAR    recommendedThemeName[128]; // doporučené jméno pro téma
    int      outputRasterFormat;        // ID výstupního rastrového formátu
                                        // generovaného mapovou službou
    COLORREF bgColor;                   // barva pozadí mapy
    // nepovinná část struktury
    // obsah nepovinné části závisí na implementaci zásuvného modulu
} WMSCONNECTION;
```

3. Jak to pracuje

Kristýna volá funkce veřejného rozhraní popsané v kapitole *Veřejné rozhraní knihovny* dle potřeby. Knihovna je uvolňována po každém volání funkce, kromě těchto případů:

- volání funkce `BOOL` **`Connect`**`(HWND hWndParent, TCHAR** formatNames, int* formatIDs, void* pSettingsStruct)` před voláním funkce `void*` **`GetConnectionSettingsStruct`**`()`.Tento případ nastává když uživatel přidává téma založené na mapové službě do zobrazení nebo mění vlastnosti tohoto tématu.
- volání funkce `BOOL` **`ReadSettingsFromStringLine`**`(TCHAR* settingsStr)` před voláním funkce `void*` **`GetConnectionSettingsStruct`**`()`.Tento případ nastává když uživatel otevírá projekt, který obsahuje téma založené na mapové službě.

3.1 Přidání vrstvy do zobrazení založené na mapové službě

Kristýna najde v registrech Windows zaregistrované typy mapových služeb. Kristýna použije hodnoty z klíčů *WMSName* a *FormatID* aby vytvořila seznam dostupných mapových služeb a naplní jimi rozbalovací nabídku v dialogovém okně pro přidání mapové služby. V tomto dialogu uživatel zvolí typ mapové služby. Po zvolení mapové služby, Kristýna přečte v registrech Windows hodnotu klíče *LibFileName*. Pokud uživatel klikne na tlačítko Spojit se s mapovou službou, Kristýna načte knihovnu a zavolá funkci Connect. Funkce Connect by měla zobrazit dialog, který dovolí uživateli spojit se s mapovou službou. Parametr pSettingsStruct ve funkci Connect má hodnotu NULL. Pokud je spojení s mapovou službou úspěšné, funkce Connect vrátí hodnotu TRUE. Poté Kristýna zavolá funkci GetConnectionSettingsStruct, alokuje paměť a uloží Vaší strukturu s nastavením pro další použití. Struktura popisující nastavení je používána funkcemi GetMap a GetDescriptionString. Používejte tuto strukturu pro sdílení dat a potřebných nastavení mezi funkcemi.

Seznam volaných funkcí během akce:

```
BOOL Connect(HWND hWndParent, TCHAR** formatNames, TCHAR** formatMINEs, int*
formatIDs, void* pSettingsStruct)
void* GetConnectionSettingsStruct()
EXTENT* GetExtent(void* pSettingsStruct)
BOOL IsQueryable(void* pSettingsStruct)
```

3.2 Vykreslování a tisk dat poskytovaných mapovou službou

Kristýna nahraje knihovnu do paměti a zavolá funkci GetMap. Funkce GetMap použije údaje ze struktury s nastavením aby se spojila s mapovou službou, přenesla obrazová data a uložila je do rastrového souboru. Požadované jméno a cesta pro rastrový soubor je specifikováno parametrem fileName. Poté uvolní knihovnu a vykreslí uložený rastrový soubor.

Seznam volaných funkcí během akce:

```
void GetMap(void* pSettingsStruct, TCHAR* fileName, HWND hWndMainFrame, EXTENT*
mapExtent, SIZE* imgSize, DWORD* pId1, volatile DWORD* pId2)
```

3.3 Zobrazení informací o mapovém prvku

Poté co uživatel klikne do zobrazení, které má první aktivní téma založeno na mapové službě, nástrojem Identifikovat  , Kristýna nahraje knihovnu do paměti a zavolá funkci GetFeatureInfo. Způsob jakým Kristýna zobrazí informace závisí na hodnotě, kterou vrátí funkce GetFeatureInfo. Podrobnější informace o možných návratových hodnotách naleznete v kapitole 4.

Seznam volaných funkcí během akce:

```
TCHAR* GetFeatureInfo(void* pSettingsStruct, TCHAR* fileName, HWND
hWndMainFrame, EXTENT* locality)
```

3.4 Zobrazení vlastností téma založeném na mapové službě

Před tím, než je zobrazen dialog Vlastnosti tématu, Kristýna zavolá funkce GetExtent, GetServerURL a GetServiceName aby byla schopná zobrazit popis mapové služby. V dialogu Vlastnosti tématu je tabulka vlastností nazvaná Mapová služba. Tato tabulka vlastností obsahuje tlačítko Změna nastavení. Uživatel může použít toto tlačítko na změnu nastavení mapové služby. Stisk tohoto tlačítka zavolá funkce Connect a následně GetConnectionSettingsStruct. Parametr pSettingsStruct ve funkci Connect obsahuje ukazatel na strukturu, která popisuje aktuální nastavení spojení. Nedovolte uživateli změnit základní parametry spojení jako URL nebo mapovou službu, ale uživatel by měl být schopen změnit jiná nastavení jako viditelnost vrstev, požadovaný výstupní rastrový formát atd.

Seznam volaných funkcí během akce:

```
BOOL Connect(HWND hWndParent, TCHAR** formatNames, TCHAR** formatMINEs, int*
formatIDs, void* pSettingsStruct)
void* GetConnectionSettingsStruct()
EXTENT* GetExtent(void* pSettingsStruct)
TCHAR* GetServerURL(void* pSettingsStruct)
TCHAR* GetServiceName(void* pSettingsStruct)
BOOL IsQueryable(void* pSettingsStruct)
```

3.5 Uložení vlastností téma založeném na mapové službě do souboru projektu

Kristýna nahraje knihovnu do paměti a zavolá funkci WriteSettingsToStringLine. Funkce WriteSettingsToStringLine použije strukturu s nastavením spojení k vytvoření jednořádkového řetězce, do kterého nastavení zapíše. Obsah řetězce závisí na Vaších potřebách. Funkce WriteSettingsToStringLine vrací ukazatel na tento řetězec. Řetězec musí být ukončen binární nulou. Kristýna uloží tento řetězec do souboru projektu jako hodnotu klíče mapServiceSettings.

Seznam volaných funkcí během akce:

```
TCHAR* WriteSettingsToStringLine(void* pSettingsStruct)
```

3.6 Načtení vlastností téma založeném na mapové službě ze souboru projektu

Kristýna nahraje knihovnu do paměti a zavolá funkci ReadSettingsFromStringLine. Funkce ReadSettingsFromStringLine použije řetězec s uloženým nastavením pro téma založené na mapové službě aby vytvořila v paměti strukturu s nastavením. Obsah řetězce byl vytvořen funkcí WriteSettingsToStringLine (viz kapitola 3.5). Řetězec je ukončen binární nulou. Funkce ReadSettingsFromStringLine vrátí hodnotu TRUE pokud byla struktura s nastavením úspěšně vytvořena, jinak vrátí hodnotu FALSE. Pokud je struktura s nastavením vytvořena úspěšně, Kristýna zavolá funkci GetConnectionSettingsStruct, vytvoří téma a uvolní knihovnu.

Seznam volaných funkcí během akce:

```
void* GetConnectionSettingsStruct()
EXTENT* GetExtent(void* pSettingsStruct)
BOOL IsQueryable(void* pSettingsStruct)
BOOL ReadSettingsFromStringLine(TCHAR* settingsStr)
```

4. Veřejné rozhraní knihovny

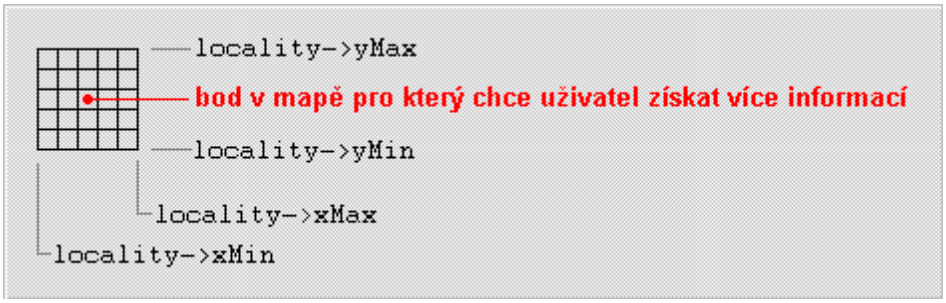
Tato kapitola popisuje veřejné rozhraní pro dynamické knihovny. Každá funkce v tomto rozhraní má přesně definované chování a je volána v situacích popsaných v sekci *Jak to pracuje*.

`BOOL` **`Connect`**`(HWND hWndParent, TCHAR** formatNames, TCHAR** formatMINEs, int* formatIDs, void* pSettingsStruct)`
Zobrazí dialog kde uživatel může specifikovat parametry spojení a spojit se s mapovou službou. Parametr hWndParent je popisovač hlavního okna Kristýny. Parametr formatNames je ukazatel na pole názvů rasterových formátů. Pole je ukončeno hodnotou NULL. Použijte tento seznam pro volbu rastrového formátu generovaného mapovou službou. Parametr formatMINEs je pole MINE řetězců pro rastrové formáty, které jsou v seznamu formatNames. Pořadí názvů formátů a MINE řetězců v těchto polích je totožné, takže první rastrový formát má název formatNames[0] a MINE řetězec MINEs[0]. Parametr formatIDs je pole identifikačních čísel rastrových formátů, které jsou v seznamu formatNames. Pořadí názvů formátů a identifikačních čísel v těchto polích je totožné, takže první rastrový formát má název formatNames[0] a indentifikační číslo formatIDs[0]. Parametr pSettingsStruct má hodnotu NULL nebo obsahuje ukazatel na strukturu, která popisuje aktuální nastavení spojení s mapovou službou (viz kapitoly 3.1 a 3.4). Tato funkce musí vyplnit nebo modifikovat strukturu popisující nastavení spojení s mapovou službou. Kristýna tuto strukturu uloží pro další použití. Pokud je spojení s mapovou službou úspěšné funkce vrátí hodnotu TRUE, jinak vrátí hodnotu FALSE.

`void*` **`GetConnectionSettingsStruct`**`()`
Vrací ukazatel na strukturu, která popisuje nastavení spojení a případně uchovává další potřebné informace.

`EXTENT*` **`GetExtent`**`(void* pSettingsStruct)`
Vrací ukazatel na rozsah dat, které tvoří mapový výstup z mapové služby v mapových jednotkách. Parametr pSettingsStruct je struktura vytvořená během navázání spojení s mapovou službou, viz kapitola 3.1.

`TCHAR*` **`GetFeatureInfo`**`(void* pSettingsStruct, TCHAR* fileName, HWND hWndMainFrame, EXTENT* locality)`
Odezvou na požadavek GetFeatureInfo je vždy soubor, který je přenesen přes internet ze serveru na klienta. Soubor obsahuje informace o mapovém prvku. Formát souboru je indikován návratovou hodnotou, viz níže. Parametr fileName je název souboru včetně cesty. Použijte tento název pro uložení přeneseného souboru. Parametr pSettingsStruct je ukazatel na strukturu, která popisuje aktuální nastavení spojení s mapovou službou (viz kapitola 3.1). Parametr hWndParent je popisovač hlavního okna Kristýny. Parametr locality definuje okolí bodu pro který uživatel chce získat informace. Parametr locality je v mapových souřadnicích a představuje čtverec 5 x 5 pixelů na uživatelově obrazovce. Ve středu tohoto čtverce je bod pro který uživatel chce získat informace.



Funkce vrátí hodnotu NULL pokud selže, jinak vrátí ukazatel na MINE řetězec, který specifikuje formát souboru s informacemi. Kristýna-GIS verze 5.x akceptuje tyto hodnoty:

- "text/plain" pro text. Tyto informace budou zobrazeny pomocí metody Report ze třídy MsgBox (Viz nápověda ke skriptovacímu jazyku Kristýny).
- "text/richtext" pro RTF text. Tyto informace budou zobrazeny pomocí metody Report ze třídy MsgBox (Viz nápověda ke skriptovacímu jazyku Kristýny).
- "text/html" for HTML text. Tyto informace budou zobrazeny pomocí metody ReportHTML ze třídy MsgBox (Viz nápověda ke skriptovacímu jazyku Kristýny).
- jiný MINE řetězec známého rastrového formátu. Tyto informace budou zobrazeny pomocí metody ShowBmp ze třídy MsgBox (Viz nápověda ke skriptovacímu jazyku Kristýny).

`DWORD` **`GetInterfaceVersion`**`()`
Vrací verzi rozhraní. Horní byte obsahuje číslo verze (1) a dolní byte číslo podverze (0).

`void` **`GetMap`**`(void* pSettingsStruct, TCHAR* fileName, HWND hWndMainFrame, EXTENT* mapExtent, SIZE* imgSize, DWORD* pId1, volatile DWORD* pId2)`
Odezvou na požadavek GetFeatureInfo je vždy soubor, který je přenesen přes internet ze serveru na klienta. Soubor představuje rastrový obrázek mapy ve výstupním formátu který je součátí poviné části struktury popisující nastavení spojení s mapovou službou. Parametr fileName je název souboru včetně cesty. Použijte tento název pro uložení přeneseného souboru. Parametr pSettingsStruct je ukazatel na strukturu, která popisuje aktuální nastavení spojení s mapovou službou (viz kapitola 3.1). Parametr hWndParent je popisovač hlavního okna Kristýny. Parametr mapExtent je požadovaný rozsah mapy v mapových jednotkách. Parametr imgSize je požadovaná velikost rastrového souboru v pixlech. . Parametry pId1 a pId2 jsou ukazatelé na číselné identifikátory. Pokud jsou hodnoty identifikátorů totožné můžete pokračovat v přenosu souboru, ale jakmile je identifikátor pId2 změněn měli by jste okamžitě zastavit přenos a uvolnit paměť a opustit funkci.

`TCHAR*` **`GetServerURL`**`(void* pSettingsStruct)`
Vrací ukazatel na řetězec, který obsahuje URL mapové služby.

`TCHAR*` **`GetServiceName`**`(void* pSettingsStruct)`
Vrací ukazatel na řetězec, který obsahuje název mapové služby.

`BOOL` **`IsQueryable`**`(void* pSettingsStruct)`
Vrací hodnotu TRUE pokud mapová služba podporuje možnost dotázat se na atributy mapových prvků. To znamená, že mapová služba je schopna reagovat na požadavek GetFeatureInfo.

`BOOL` **`ReadSettingsFromStringLine`**`(TCHAR* settingsStr)`
Během otevírání projektu přečte nastavení z jednořádkového řetězce.

`TCHAR*` **`WriteSettingsToStringLine`**`(void* pSettingsStruct)`
Při ukládání projektu do souboru zapíše nastavení do jednořádkového řetězce. Vrací ukazatel na tento řetězec.

4.1 Minimální funkční požadavky

Minimální požadavek je vykreslení dat poskytovaných mapovou službou v zobrazení a tisk těchto dat na tiskárně. K vytvoření fungující knihovny, která toto splňuje musíte implementovat následující funkce s popsaným chováním:

`BOOL` **`Connect`**`(HWND hWndParent, TCHAR** formatNames, TCHAR** formatMINEs, int* formatIDs, void* pSettingsStruct)`
Dovolí uživatelům spojit se s mapovou službou a konfigurovat ji. Vytvoří a vyplní strukturu sloužící k uchování parametrů spojení.

`void*` **`GetConnectionSettingsStruct`**`()`
Vrátí ukazatel na strukturu, která uchovává parametry spojení.

`EXTENT*` **`GetExtent`**`(void* pSettingsStruct)`
Vrátí ukazatel na rozsah dat, které tvoří mapový výstup z mapové služby.

`TCHAR*` **`GetFeatureInfo`**`(void* pSettingsStruct, TCHAR* fileName, HWND hWndMainFrame, EXTENT* locality)`
Vrátí hodnotu NULL.

`DWORD` **`GetInterfaceVersion`**`()`
Vrátí verzi rozhraní. Horní byte obsahuje číslo verze (1) a dolní byte číslo podverze (0).

`void` **`GetMap`**`(void* pSettingsStruct, TCHAR* fileName, HWND hWndMainFrame, EXTENT* mapExtent, SIZE* imgSize, DWORD* pId1, volatile DWORD* pId2)`
Uloží mapový výstup do souboru.

`TCHAR*` **`GetServerURL`**`(void* pSettingsStruct)`
Vrátí ukazatel na řetězec, který obsahuje URL mapové služby.

`TCHAR*` **`GetServiceName`**`(void* pSettingsStruct)`
Vrátí ukazatel na řetězec, který obsahuje název mapové služby.

`BOOL` **`IsQueryable`**`(void* pSettingsStruct)`
Vrátí FALSE.

`BOOL` **`ReadSettingsFromStringLine`**`(TCHAR* settingsStr)`
Při otevírání projektu přečte nastavení z jednořádkového řetězce.

`TCHAR*` **`WriteSettingsToStringLine`**`(void* pSettingsStruct)`
Zapíše nastavení do jednořádkového řetězce.

5. Instalace Vaší vlastní knihovny

Doporučujeme vytvořit instalační program pro instalaci knihovny. Instalační program musí umístit knihovnu do podadresáře dlls v instalačním adresáři Kristýny. Instalační program musí také zapsat následující informace do registrů Windows:

Klíč:
HKEY_LOCAL_MACHINE\Software\Christine-GIS\5.x\WMS\[ID typu mapové služby]

Hodnoty:

Název	Hodnota	Typ	Max. délka (byte)	
LibFileName	[library.dll]	REG_SZ	255	povinné
WMSName	Název typu mapové služby	REG_SZ	255	povinné
FormatID	[ID typu mapové služby]	REG_DWORD	4	povinné

Příklad pro OGC WMS:

Klíč:
HKEY_LOCAL_MACHINE\Software\Christine-GIS\5.x\WMS\1002

Hodnoty:

Název	Hodnota
LibFileName	ogcwms.dll
WMSName	OGC Web Map Service
FormatID	1002

Poznámka: Hodnota *FormatID* musí být stejná jako název klíče.

5.1 Identifikační číslo typu mapové služby (ID)

Pokud chcete vytvářet vlastní knihovnu pro práci s daty poskytovanými internetovou/intranetovou mapovou službou, ověřte si na webové stránce Kristýny volná identifikační čísla typů mapových služeb. Poté vyberte volné ID, kontaktujte nás a my zveřejníme Vámi zvolené ID na webu Kristýny abychom zabránili problémům s duplicitou. Identifikační číslo formátu je používáno interně Kristýnou a těmito metodami ve skriptovacím jazyce Kristýny: GetType(Number nType) ze třídy Theme, ExportToBmp(String sFullFileName, Number nType, Rect extent, Number nWidth, Number nHeight, Bool bCreateWorldFile) a AddTheme(String sFullFileName, Number nType) ze třídy View, ExportToVectorFormat(String sFullFileName, Number nFormatID, Bool bUseDefaultSettings) ze třídy FTheme, Export(String sFullFileName, Number nOutputType) ze třídy DSCTheme, ShowBmp(String sFullFileName, Number nDelay, Bool bFrame, Number nType) ze třídy MsgBox.

Doporučené rozsahy identifikačních čísle jsou následující:

0 - 1000 pro rastrová a vektorová souborová data (typicky uložena na souborovém systému), rozsah 0 - 100 je rezervován pro Kristýnu-GIS, 101 - 1000 je k dispozici pro vývojáře třetích stran

1001 - 2000 pro rastrová a vektorová data z webovských služeb (typicky generována mapovými servery a zaslána Kristýně), rozsah 1001 - 1100 je rezervován pro Kristýnu-GIS, 1101 - 2000 je k dispozici pro vývojáře třetích stran

2001 - 3000 pro rastrová a vektorová data uložena v geodatabázi (pro budoucí využití, nyní nepodporováno), rozsah 2001 - 2100 je rezervován pro Kristýnu-GIS, 2101 - 3000 je k dispozici pro vývojáře třetích stran

6. Knihovny poskytované s Kristýnou

S Kristýnou-GIS verze 5.x jsou poskytovány níže uvedené knihovny, které mají implementováno toto rozhraní. Tyto knihovny jsou umístěny v podadresáři dlls v instalačním adresáři Kristýny.

Typ mapové služby:	**OGC Web Map Service**
Jméno knihovny:	ogcwms.dll
ID:	1002
Verze rozhraní:	1.0

7. Příklad

Nabízíme Vám zdrojový kód prázdné kostry pro dynamickou knihovnu používající toto rozhraní. Kód je v jazyce C++ a je testován ve vývojovém prostředí Microsoft Visual Studio 2012.

7.1 Jak příklad zkompilovat

Tato krátká kapitola popisuje kroky vedoucí nutné ke kompilaci zdrojového kódu z kapitoly 7.2 v prostředí Microsoft Visual Studio 2012. Jednoduše následujte instrukce uvedené níže.

- Spusťte Microsoft Visual Studio 2012.
- Z nabídky File zvolte New a poté vyberte příkaz Project. Z kategorií projetků vyberte Win32 v sekci Visual C++. Z instalovaných šablon zvolte Win32 Project. Zvolte umístění pro nový projekt a poté vepište jméno projektu a stiskněte tlačítko OK. V následujícím průvodci zvolte volbu „DLL" a stiskněte tlačítko Finish.
- Je vytvořen projekt, který obsahuje soubor dllmain.cpp.
- Zkopírujte zdrojový kód z kapitoly 7.2. do souboru dllmain.cpp.
- Z nabídky Project zvolte příkaz projectName Properties (Alt + F7). Vyberte sekci Configuration Properties/General a nastavte parametr „Character Set“ na hodnotu „Not Set“. Nyní vyberte sekci Configuration Properties/C/C++/Preprocesor a přidejte do Preprocessor Definitions hodnotu „_CRT_SECURE_NO_WARNINGS“.
- Z nabídky Build zvolte příkaz Build Solution (F7).

7.2 Zdrojový kód

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

typedef struct _EXTENT {      // rozsah dat
   double xMin;
   double yMin;
   double xMax;
   double yMax;
   double zMin;
   double zMax;
   double mMin;
   double mMax;
} EXTENT;

// struktura uchovávájící parametry spojení popřípadě veškerá jiná potřebná
// data (dále jen "Struktura")
typedef struct _WMSCONNECTION {
        // povinná část struktury
        DWORD   dwSize;                     // velikost této struktury (včetně nepovinné části)
        TCHAR   recommendedThemeName[128];  // doporučené jméno pro téma
        int     outputRasterFormat;         // ID výstupního rastrového formátu
                                            // generovaného mapovou službou
        COLORREF bgColor;                   // barva pozadí mapy
        // nepovinná část struktury - závisí na Vašich potřebách
        // ...
        // zde deklarujte Vaše vlastní členy struktury
        // ...
        EXTENT  extent;
        TCHAR   buffer[1024];
} WMSCONNECTION;

HINSTANCE hInstanceDll;
static DWORD   dwTlsIndex; // adresa sdílené paměti
```

```
BOOL WINAPI DllMain(HINSTANCE hInstDll, DWORD fdwReason, LPVOID lpReserved)
{
    LPVOID lpvData;
       // Podle důvodu volání proveďte patřičnou akci.
       switch(fdwReason)
       {
              case DLL_PROCESS_ATTACH: {
                     // Inicializace pro každý nový proces. Vrací FALSE pokud natažení DLL selže.

                     // Uložte si popisovač instance knihovny
                     hInstanceDll = hInstDll;

                     // alokuji TLS index
                     if ((dwTlsIndex = TlsAlloc()) == TLS_OUT_OF_INDEXES)
                            return FALSE;
                     // žádný break: Inicializuji index pro první vlákno
              }
              case DLL_THREAD_ATTACH:
                     // Inicializace pro vlákno.
                     // Inicializuji TLS index pro toto vlákno
                     lpvData = (LPVOID)new BYTE[sizeof(WMSCONNECTION)];
                     if (lpvData == NULL) return FALSE;
                     if (!TlsSetValue(dwTlsIndex, lpvData)) {
                            delete[] lpvData;
                            return FALSE;
                     }
                     break;

              case DLL_THREAD_DETACH:
                     // Úklid pro vlákno.
                     // Uvolním alokovanou paměť pro toto vlákno
                     lpvData = TlsGetValue(dwTlsIndex);
                     if (lpvData != NULL) {
                            BYTE* wmsConnection = (BYTE*)lpvData;
                            if (wmsConnection) delete[] wmsConnection;
                     }
                     break;

              case DLL_PROCESS_DETACH:
                     // Úklid pro proces.
                     lpvData = TlsGetValue(dwTlsIndex);
                     if (lpvData != NULL) {
                            BYTE* wmsConnection = (BYTE*)lpvData;
                            if (wmsConnection) delete[] wmsConnection;
                     }
                     // Uvolním TLS index.
                     TlsFree(dwTlsIndex);
                     break;
       }
    return TRUE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl Connect(HWND hWndParent, TCHAR** formatNames,
TCHAR** formatMINEs, int* formatIDs, void* pSettingsStruct)
{
       BYTE* lpvData = (BYTE*)TlsGetValue(dwTlsIndex);
       if (!lpvData) {
              lpvData = new BYTE[sizeof(WMSCONNECTION)];
              if (!lpvData) return FALSE;
              if (!TlsSetValue(dwTlsIndex, lpvData)) {
                     delete[] lpvData;
                     return FALSE;
              }
       }
       // kopíruji Strukturu
       if (pSettingsStruct) {
              delete[] lpvData;
              lpvData = new BYTE[((DWORD*)pSettingsStruct)[0]];
              if (!lpvData) return FALSE;
              memcpy(lpvData, pSettingsStruct, ((DWORD*)pSettingsStruct)[0]);
              if (!TlsSetValue(dwTlsIndex, lpvData)) {
                     delete[] lpvData;
                     return FALSE;
              }
       }
       // Inicializuji Strukturu
       WMSCONNECTION* wmsConnection = (WMSCONNECTION*)lpvData;
       wmsConnection->dwSize = sizeof(WMSCONNECTION);
       lstrcpy(wmsConnection->recommendedThemeName, "aThemeName");
```

```
	wmsConnection->outputRasterFormat = 7;		// PNG
	wmsConnection->bgColor = RGB(255,255,255);	// bílá barva

	// Zobrazí dialog, který dovolí uživateli spojit se s mapovou službou
	// nebo modifikuje spojení a nastavení mapové služby.
	// Pokud je spojení úspěšné, odpojte se od mapové služby,
	// uložte parametry spojení a jakákoliv jiná potřebná data
	// do Struktury a vraťte hodnotu TRUE. Pokud je spojení neúspěšné
	// vraťte hodnotu FALSE;
	// ...
	// zde vložte Váš kód
	// ...

	return TRUE;
}

extern "C" __declspec( dllexport ) void* __cdecl GetConnectionSettingsStruct()
{
	// vrací ukazatel na Strukturu
	// Kristýna ji uloží do paměti pro další použití
	return TlsGetValue(dwTlsIndex);
}

extern "C" __declspec( dllexport ) EXTENT* __cdecl GetExtent(void* pSettingsStruct)
{
	// ukazatel na Strukturu
	WMSCONNECTION* wmsConnection = (WMSCONNECTION*)pSettingsStruct;

	memset(&(wmsConnection->extent), 0, sizeof(EXTENT));

	// vyplňte členy xMin, yMin, xMax a yMax
	// ...
	// zde vložte Váš kód
	// ...

	return &(wmsConnection->extent);
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl GetFeatureInfo(void* pSettingsStruct, TCHAR*
fileName, HWND hWndMainFrame, EXTENT* locality)
{
	BYTE* lpvData = (BYTE*)TlsGetValue(dwTlsIndex);
	if (lpvData) delete[] lpvData;
	// kopíruji Strukturu
	lpvData = new BYTE[((DWORD*)pSettingsStruct)[0]];
	if (!lpvData) return NULL;
	memcpy(lpvData, pSettingsStruct, ((DWORD*)pSettingsStruct)[0]);
	if (!TlsSetValue(dwTlsIndex, lpvData)) {
		delete[] lpvData;
		return NULL;
	}
	WMSCONNECTION* wmsConnection = (WMSCONNECTION*)lpvData;

	// Souřadnice bodu, pro který uživatel chce získat informace
	double x = locality->xMin + ((locality->xMax - locality->xMin) / 2);
	double y = locality->yMin + ((locality->yMax - locality->yMin) / 2);

	// Získejte informace o mapovém prvku a uložte je do souboru s názvem fileName
	// ...
	// zde vložte Váš kód
	// ...

	// Pokud je vše úspěšně hotovo, vraťte ukazatel na MINE řetězec,
	// který popisuje formát souboru, jinak vraťte hodnotu NULL

	// return wmsConnection->buffer;
	return NULL;
}

extern "C" __declspec( dllexport ) DWORD __cdecl GetInterfaceVersion()
{
	// vraťte verzi rozhraní (1.0)
	return MAKELONG(0, 1);
}
```

```
extern "C" __declspec( dllexport ) void __cdecl GetMap(void* pSettingsStruct, TCHAR* fileName, HWND
hWndMainFrame, EXTENT* mapExtent, SIZE* imgSize, DWORD* pId1, volatile DWORD* pId2)
{
	BYTE* lpvData = (BYTE*)TlsGetValue(dwTlsIndex);
	if (lpvData) delete[] lpvData;
	// kopíruji Strukturu
	lpvData = new BYTE[((DWORD*)pSettingsStruct)[0]];
	if (!lpvData) return;
	memcpy(lpvData, pSettingsStruct, ((DWORD*)pSettingsStruct)[0]);
	if (!TlsSetValue(dwTlsIndex, lpvData)) {
		delete[] lpvData;
		return;
	}
	WMSCONNECTION* wmsConnection = (WMSCONNECTION*)lpvData;

	// Funkce běží dokud *pId1 je rovno *pId2
	if (*pId1 != *pId2) return;

	// ...
	// zde vložte Váš kód
	// ...
	// Prosím nezapomeňte na hodnotu pId2. Kontrolujte tuto hodnotu
	// tak často jak je to možné a rozumné. Pokud je hodnota změněna
	// okamžitě uvolněte paměť a opusťte funkci.
	// ...

}

extern "C" __declspec( dllexport ) TCHAR* __cdecl GetServiceName(void* pSettingsStruct)
{
	BYTE* lpvData = (BYTE*)TlsGetValue(dwTlsIndex);
	if (lpvData) delete[] lpvData;
	// kopíruji Strukturu
	lpvData = new BYTE[((DWORD*)pSettingsStruct)[0]];
	if (!lpvData) return NULL;
	memcpy(lpvData, pSettingsStruct, ((DWORD*)pSettingsStruct)[0]);
	if (!TlsSetValue(dwTlsIndex, lpvData)) {
		delete[] lpvData;
		return NULL;
	}
	WMSCONNECTION* wmsConnection = (WMSCONNECTION*)lpvData;
	wmsConnection->buffer[0] = '\0';

	// Naplňte wmsConnection->buffer názvem mapové služby
	// ...
	// zde vložte Váš kód
	// ...

	return wmsConnection->buffer;
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl GetServerURL(void* pSettingsStruct)
{
	BYTE* lpvData = (BYTE*)TlsGetValue(dwTlsIndex);
	if (lpvData) delete[] lpvData;
	// kopíruji Strukturu
	lpvData = new BYTE[((DWORD*)pSettingsStruct)[0]];
	if (!lpvData) return NULL;
	memcpy(lpvData, pSettingsStruct, ((DWORD*)pSettingsStruct)[0]);
	if (!TlsSetValue(dwTlsIndex, lpvData)) {
		delete[] lpvData;
		return NULL;
	}
	WMSCONNECTION* wmsConnection = (WMSCONNECTION*)lpvData;
	wmsConnection->buffer[0] = '\0';

	// Zapište URL mapové služby do wmsConnection->buffer
	// ...
	// zde vložte Váš kód
	// ...

	return wmsConnection->buffer;
}
```

```
extern "C" __declspec( dllexport ) BOOL __cdecl IsQueryable(void* pSettingsStruct)
{
	BYTE* lpvData = (BYTE*)TlsGetValue(dwTlsIndex);
	if (lpvData) delete[] lpvData;
	// kopíruji Strukturu
	lpvData = new BYTE[((DWORD*)pSettingsStruct)[0]];
	if (!lpvData) return FALSE;
	memcpy(lpvData, pSettingsStruct, ((DWORD*)pSettingsStruct)[0]);
	if (!TlsSetValue(dwTlsIndex, lpvData)) {
		delete[] lpvData;
		return FALSE;
	}
	WMSCONNECTION* wmsConnection = (WMSCONNECTION*)lpvData;

	// Zjistěte zda mapová služba umožňuje dotazovat se na informace o mapových prvcích
	// ...
	// zde vložte Váš kód
	// ...

	// pokud mapová služba umožňuje dotazovat se na informace o mapových prvcích vraťte hodnotu
TRUE,
	// jinak vraťte hodnotu FALSE
	return FALSE;
}

extern "C" __declspec( dllexport ) BOOL __cdecl ReadSettingsFromStringLine(TCHAR* settingsStr)
{
	BYTE* lpvData = (BYTE*)TlsGetValue(dwTlsIndex);
	if (lpvData) delete[] lpvData;
	// vytvořím Strukturu
	lpvData = new BYTE[sizeof(WMSCONNECTION)];
	if (!lpvData) return FALSE;
	if (!TlsSetValue(dwTlsIndex, lpvData)) {
		delete[] lpvData;
		return FALSE;
	}
	// Inicializuji Strukturu
	WMSCONNECTION* wmsConnection = (WMSCONNECTION*)lpvData;
	wmsConnection->dwSize = sizeof(WMSCONNECTION);
	lstrcpy(wmsConnection->recommendedThemeName, "aThemeName");
	wmsConnection->outputRasterFormat = 7;              // PNG
	wmsConnection->bgColor = RGB(255,255,255);  // bílá barva

	// naplním Strukturu
	// ...
	// zde vložte Váš kód
	// ...

	// pokud byla struktura úspěšně vytvořena a naplněna vraťte hodnotu TRUE
	return TRUE;
}

extern "C" __declspec( dllexport ) TCHAR* __cdecl WriteSettingsToStringLine(void* pSettingsStruct)
{
	BYTE* lpvData = (BYTE*)TlsGetValue(dwTlsIndex);
	if (lpvData) delete[] lpvData;
	// kopíruji Strukturu
	lpvData = new BYTE[((DWORD*)pSettingsStruct)[0]];
	if (!lpvData) return NULL;
	memcpy(lpvData, pSettingsStruct, ((DWORD*)pSettingsStruct)[0]);
	if (!TlsSetValue(dwTlsIndex, lpvData)) {
		delete[] lpvData;
		return NULL;
	}
	WMSCONNECTION* wmsConnection = (WMSCONNECTION*)lpvData;
	wmsConnection->buffer[0] = '\0';

	// Naplňte wmsConnection->buffer
	// ...
	// zde vložte Váš kód
	// ...

	return wmsConnection->buffer;
}
```