Modale GeoMedia® .NET-Customcommands mit VB2010

1. GeoMedia Customcommand unter *Visual Basic 2010* erstellen

1.1. *Visual Basic 2010*-Projektvorlage *Modaler Customcommand VB2010*

Der *GeoMedia Commandwizard* steht bisher für *Visual Studio 2010* nicht zur Verfügung. Die Codestruktur muss in *Visual Studio 2010* deshalb von einem alten Customcommand-Projekt übernommen und angepasst werden oder für modale Customcommands mit Hilfe der *Visual Studio 2010*-Projektvorlage *Modaler Customcommand VB2010* erstellt werden.

Die im Folgenden beschriebenen codeseitigen Anforderungen und Einstellungen sind in der *Visual Basic 2010*-Projektvorlage *Modaler Customcommand VB2010* berücksichtigt. Die Vorlage muss dazu als zip-Datei in das Vorlagenverzeichnis *<Eigene Dateien>\Visual Studio 2010\Templates\ProjectTemplates\Visual Basic\[GeoMedia]* abgelegt werden und steht damit im IDE-Dialogfenster *Neues Projekt* unter den *Visual Basic*-Vorlagen zur Verfügung. Bei Verwendung der Vorlage muss alleinig die Debugging-Einstellung (vgl. 1.3.3) manuell vorgenommen werden.

1.2. Zwingende Einstellungen im *Visual Basic 2010*-Projekt

1.2.1. Codestruktur modaler Customcommand

Die Klasse eines modalen Customcommands muss mindestens die Methoden `Initialize`, `Activate` und `Terminate` besitzen. Abhängig von den Aktivierungsbedingungen muss zusätzlich die Methode `CanEnable` und optional die Methode `Help` implementiert werden.

1.2.2. Klasse und Member müssen öffentlich sein (`Public`)

Die Klasse für den Customcommand muss als öffentliche Klasse deklariert werden. Ebenso die für den GeoMedia Commandserver relevanten Methoden (vgl. 1.2.1).

1.2.3. COM-Sichtbarkeit

Die Klasse muss für den Aufruf durch den GeoMedia Commandserver COM-sichtbar gemacht werden. Dies kann auf zwei Arten erfolgen: Entweder projektglobal über die Option *Assembly COM-sichtbar machen* unter den *Projekteinstellungen > Anwendung > Assemblyinformationen* oder mit Hilfe des `ComVisible`-Attributs direkt im Codemodul der Klasse. Für eine Klasse hat die `Class`-Deklaration dann die Form

```
<System.Runtime.InteropServices.ComVisible(True)>
Public Class ModalCmd
```

1.2.4. COM-kompatible ProgID

Die ProgID wird ohne explizite Angaben automatisch in der Form `<Namespace>.<Classname>` generiert. Diese ProgID muss für die Verwendung mit dem GeoMedia Commandserver COM-kompatibel sein, d.h. insbesondere sie darf maximal 39 Zeichen lang sein und keine Sonderzeichen enthalten. Um Kompatibilitätsprobleme durch die implizit generierte ProgID im Vorneherein zu verhindern, empfiehlt sich die ProgID explizit zu definieren (vgl. dazu 1.3.1).

1.2.5. COM-Ressourcen freigeben

Die im Projekt erstellten und benutzten COM-Objekte, also die Objekte aus den GeoMedia-Programmbibliotheken, müssen bei der Verweisfreigabe mit dem Marshalling-Service[1] freigegeben werden. Zudem sollte die .NET Garbage Collection explizit aufgerufen werden. Nicht sauber freigegebene Ressourcen können GeoMedia zum Absturz bringen oder das saubere Beenden des Windows-Prozesses verhindern.[2] Am Beispiel der Freigabe der `Application`-

[1] Die `Marshal`-Klasse stellt Methoden für den Datenaustausch und die Kommunikation zwischen verwaltetem und nicht verwaltetem Code, so genanntes Marshalling, bereit.

[2] In einem solchen Fall muss das Ende des Prozesses *GeoMedia.exe* mit dem *Windows Task-Manager* erzwungen werden. Das Ganze verhält sich aber ab und an auch mit den vorgenommenen Freigaben

Instanz in der Methode `Terminate` ist die Form einer solchen Ressourcenfreigabe ersichtlich:

```
If Not IsNothing(objGMApp) Then _
   System.Runtime.InteropServices.Marshal.ReleaseComObject(objGMApp)
objGMApp = Nothing
GC.Collect()
GC.WaitForPendingFinalizers()
```

1.3. Optimierende Einstellungen im *Visual Basic 2010*-Projekt

Die in diesem Abschnitt aufgeführten Einstellungen sind technisch nicht in jedem Fall zwingend vorzunehmen. Um den Code zu optimieren, empfiehlt es sich aber diese Einstellungen zu beachten. In der *Visual Basic 2010*-Projektvorlage *Modaler Customcommand VB2010* sind sie bereits berücksichtigt.

1.3.1. ProgID explizit angeben

Die ProgID der Customcommand-Klasse kann mit Hilfe des `ProdId`-Attributs explizit angegeben werden. Zusammen mit dem `ComVisible`-Attribut (vgl. 1.2.3) hat die `Class`-Deklaration dann die Form

```
<System.Runtime.InteropServices.ProgId("MyGMCommands.ModalCmd")>
<System.Runtime.InteropServices.ComVisible(True)>
Public Class ModalCmd
```

Beachten Sie, dass eine COM-ProgID auf dem Zielrechner systemweit eindeutig sein muss! Gebräuchlich setzt sich die ProgID aus dem Kurznamen des Produkts und dem Klassennamen, getrennt durch ein Punktzeichen und evtl. um eine Versionsnummer ergänzt, zusammen, also `<Produkt>.<Klassennamen>` oder auch `<Produkt>.<Klassennamen>.<Version>`.

1.3.2. Signieren der Assembly mit starken Namen

Die Signierung der Assembly mit einem eindeutigen Namen verleiht dieser eine global eindeutige Identität. Optional können dazu Verschlüsselung und Zertifikate verwendet werden um höhere Anforderungen an die Sicherheit beim Einsatz der Assembly zu stellen.

Im unkompliziertesten Fall wird die Assembly einfach unter *Projekteinstellungen* > *Signierung* signiert, indem die Option *Assembly signieren* aktiviert wird. Unter der Einstellung *Schlüsseldatei mit starkem Namen auswählen* kann dann mit dem Eintrag *<Neu…>* eine Schlüsseldatei mit oder ohne Kennwortschutz erstellt werden.[3]

1.3.3. Debuggen von Klassenbibliotheken – auch mit der Express-Edition

Mit der *Visual Studio 2010 Professional*-Edition können Sie einen Customcommand bequem über die *Startaktion* unter *Projekteinstellungen* > *Debuggen* debuggen. Wählen Sie dort die Option *Externes Programm starten* und geben Sie als Argument die GeoMedia-Programmdatei[4] an.

Mit der *Visual Basic 2010 Express*-Edition steht diese Bedienungsoption leider nicht direkt zur Verfügung. Jedoch kann die Datei *<Projekt>.vbproj.user* manuell erweitert werden um dasselbe Debug-Startverhalten zu ermöglichen. Ergänzen Sie dazu die *vbproj.user*-Datei mit folgendem Abschnitt:

---

instabil und ist teilweise intransparent. Zum Beispiel führt der Zugriff auf die Eigenschaft `ActiveWindow` des `Application`-Objekts in der Methode `CanEnable` nach bisherigen Tests beim Beenden von GeoMedia konsequent zum Absturz.

[3] Gleichwertige Operationen können mit den .NET-Tools *Strong Name* und *Assembly Linker* vorgenommen werden. Die Dokumentation dazu findet sich in der MSDN-Dokumentation.

[4] standardmässig *C:\Programme\GeoMedia[Professional]\Program\GeoMedia.exe*

```
<PropertyGroup>
   <StartAction>Program</StartAction>
   <StartProgram>
      C:\Programme\GeoMedia Professional\Program\GeoMedia.exe
   </StartProgram>
</PropertyGroup>
```

In der *Visual Studio 2010*-Projektvorlage *Modaler Customcommand VB2010* wird im Unterverzeichnis *bin* eine Vorlage für diese Einstellung mitgeliefert. Kopieren Sie den Inhalt dieser Datei oder gerade die ganze Datei nach *<Projekt>.vbproj.user* im Projektstammverzeichnis. Die Angabe des GeoMedia-Installationsverzeichnisses ist gegebenenfalls entsprechend anzupassen.

1.3.4. *RegAsm*-Registrierung durch die IDE

Die notwendige COM-Registrierung mit dem *RegAsm*-Tool kann mit der *Visual Studio Professional*-Edition bequem in der IDE vorgenommen werden. Aktivieren Sie dazu einfach die Option *Für COM-Interop registrieren* unter *Projekteinstellungen* > *Kompilieren*. Dadurch wird bei jedem Kompilieren der Assembly die COM-Registrierung mit den Optionen `/tlb` und `/codebase` vorgenommen (vgl. 2.1).

Mit der *Visual Basic 2010 Express*-Edition steht diese Bedienungsoption leider nicht zur Verfügung. Die COM-Registrierung muss immer manuell vorgenommen werden. Am Einfachsten erfolgt dies mit Hilfe einer Batch-Datei. In der *Visual Studio 2010*-Projektvorlage *Modaler Customcommand VB2010* wird eine solche Batch-Datei bereits mitgeliefert.

2. GeoMedia Customcommand registrieren

Die Angaben in diesem Abschnitt sind nicht nur für modale Customcommands gültig. Der Registriervorgang ist unabhängig vom Commandtyp.

Ein Customcommand wird durch die GeoMedia-Registrierung dem GeoMedia Commandserver bekannt gemacht. .NET-Customcommands müssen zudem für den COM-Zugriff unter Windows mit dem *RegAsm*-Tool verfügbar gemacht werden.[5]

2.1. Customcommand mit *RegAsm*-Tool COM-verfügbar machen

Die Registrierung mit dem *RegAsm*-Tool muss zumindest mit der Option `/codebase` erfolgen.[6] Zudem kann zum Export der Assembly-Metadaten nach COM die Option `/tlb` verwendet werden. Damit wird eine Typenbibliothek erzeugt und registriert. Ein Registrieraufruf sieht dann folgendermassen aus:

```
RegAsm MyDotNETGMCmd.dll /tlb /codebase
```

Die einzusetzende *RegAsm*-Version ist abhängig vom gewählten Zielframework (*Projekteinstellungen* > *Kompilieren* > *Erweiterte Kompilieroptionen* > *Zielframework*). Für die .NET Framework-Versionen ab 2.0 bis und mit 3.5 ist die *RegAsm*-Version aus dem Verzeichnis *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727* zu verwenden, für die .NET Framework-Version 4.0 die *RegAsm*-Version aus dem Verzeichnis *C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319*.

Die Registrierung mit dem *RegAsm*-Tool sollte *vor* der GeoMedia-Registrierung erfolgen. Für die Registrierung als Usercommand mit *InstallUsrCmd* (vgl. 2.2.2) ist dies zwingend. Während der Entwicklung eines Customcommands ist es abhängig von der eingesetzten *Visual*

[5] Mit dem beschriebenen Vorgehen betreffend der Projektkonfiguration und des Registrierungsvorgangs wird die .NET-Klasse des Customcommands für den COM-Zugriff *verfügbar* gemacht. Um ein „echte" COM-Klasse/-Bibliothek unter .NET zu erstellen sind weitere Einstellungen und Anforderungen an die Klasse zu beachten. Weitere Informationen dazu finden sich in der MSDN-Dokumentation und entsprechender Fachliteratur. In der *Visual Studio 2010 Professional*-Edition steht eine Projektvorlage für COM-Klassen zur Verfügung.

[6] Dies ist insbesondere notwendig, wenn sich eine Assembly weder im *Global Assembly Cache (GAC)* noch im Anwendungsverzeichnis der Hauptanwendung befindet. Es empfiehlt sich für diese Registrierungsoption die Assembly mit einem starken Namen zu signieren (vgl. 1.3.2).

Studio-Edition (*Professional*, *Express*, …), inwieweit die Registrierung mit dem *RegAsm*-Tool direkt von der IDE erledigt wird (vgl. 1.3.4). Auf einem produktiven Zielrechner muss die Registrierung mit dem *RegAsm*-Tool immer explizit vorgenommen werden.

Der Deregistrierungsvorgang erfolgt analog zur Registrierung mit der Option `/u`:

```
RegAsm /u MyDotNETGMCmd.dll /tlb /codebase
```

2.2. GeoMedia-Registrierung eines Customcommands

Für die GeoMedia-Registrierung gibt es zwei mögliche Varianten: die Registrierung als Applikationscommand (für alle Accounts auf dem Zielrechner) und die Registrierung als Usercommand (nur für den Account). Weitere Details zu den beiden Registrierungsvarianten finden sich auch in der *GeoMedia-Hilfe für den Befehls-Assistenten* (Einträge *Install Application Command Tool* und *Install User Command Tool*).

2.2.1. Registrierung als Applikationscommand

Die Registrierung als Applikationscommand erfolgt mit dem GeoMedia-Tool *InstallAppCmd*. Dieses befindet sich im Unterverzeichnis *Program* des GeoMedia-Installationsverzeichnisses[7]. Diese Registrierung muss mit Admin-Rechten durchgeführt werden.

Ein solcher Registrieraufruf hat die Form

```
InstallAppCmd /cmdset MyGMCommands /loc %cd% /dll %cd%
```

Die Option `/cmdset` gibt den frei wählbaren Namen der Commandsammlung an. Die Option `/loc` gibt den Verzeichnispfad der XML-Konfigurationsdatei des Customcommands an. Die Option `/dll` den Verzeichnispfad der Programmdatei (Assembly) des Customcommands.

Befinden sich XML-Konfigurationsdatei und Programmdatei im selben Verzeichnis und wird der Registrieraufruf mit Hilfe einer Batch-Datei in diesem Verzeichnis ausgeführt, wird am Einfachsten wie im obigen Beispiel die Variable `%cd%` eingesetzt, die gerade den Verzeichnisnamen enthält.

Ein Registrieraufruf mit `InstallAppCmd` registriert alle in den angegebenen Verzeichnissen (Optionen `/loc` und `/dll`) befindlichen Customcommands.

2.2.2. Registrierung als Usercommand

Die Registrierung als Usercommand erfolgt mit dem GeoMedia-Tool *InstallUsrCmd*. Dieses befindet sich ebenfalls im Unterverzeichnis *Program* des GeoMedia-Installationsverzeichnisses[7]. Ein solcher Registrieraufruf hat die Form

```
InstallUsrCmd MyDotNETGMCmd.dll MyDotNETGMCmd.ini
```

2.2.3. Deregistrierung von Customcommands

Die Deregistrierung der Customcommands erfolgt für beide Registrierungsvarianten mit der Option `/u`:

```
InstallAppCmd /u /cmdset MyGMCommands /loc %cd% /dll %cd%
```

bzw.

```
InstallUsrCmd /u MyDotNETGMCmd.dll MyDotNETGMCmd.ini
```

Mit der Aufhebung der GeoMedia-Registrierung sollte auch die Deregistrierung mit dem *RegAsm*-Tool erfolgen. Für Usercommands ist die Deregistrierung mit dem *RegAsm*-Tool zwingend *vor* der Deregistrierung mit *InstallUsrCmd* durchzuführen.

[7] standardmässig *C:\Programme\GeoMedia[Professional]*

3. XML-/ini-Konfigurationsdatei für modale Customcommands

Die Parameter der Konfigurationsdatei (XML- oder ini-Datei) eines modalen Customcommands sind relativ leicht manuell anzupassen. Für andere Commandtypen empfiehlt sich die Konfigurationsdateien mit Hilfe des *GeoMedia Commandwizard*s zu erstellen.

Die wesentlichen Parameter in den Konfigurationsdateien eines modalen Customcommand sind `ProgID`, `EnableMask`, `Description` und `ToolTip`. Für die Registrierung als Applikationscommand (XML-Konfigurationsdatei) zudem der Parameter `DllName` und optional der Parameter `BitmapRootFileName`, sofern Toolbar-Bitmaps eingesetzt werden. Für die Registrierung als Usercommand (ini-Konfigurationsdatei) ist die Abschnittsbezeichnung anzupassen.[8] Die übrigen Parameter bleiben für einen modalen Customcommand unverändert (vgl. Beispiele).

Die Parameter sind auch in der *GeoMedia-Hilfe für den Befehls-Assistenten* beschrieben. Vorlagen für die Konfigurationsdateien werden mit der *Visual Studio 2010*-Projektvorlage *Modaler Customcommand VB2010* mitgeliefert.

3.1. Beispiel einer XML-Konfigurationsdatei für Applikationscommands

Der Inhalt einer XML-Konfigurationsdatei für die Registrierung als Applikationscommand sieht ohne XML-Header exemplarisch folgendermassen aus:

```
<ApplicationCommand xmlns="http://www.intergraph.com/GeoMedia/appcmd">
   <ProgID>MyGMCommands.ModalCmd</ProgID>
   <DllName>ModalCmd.dll</DllName>
   <BitmapRootFileName>ModalCmd</BitmapRootFileName>
   <Description>Beschreibung der Funktion des Customcommands</Description>
   <Tooltip>Kurzbeschreibung Customcommand</Tooltip>
   <EnableMask>1</EnableMask>
   <IsModal>1</IsModal>
   <ModelessCommandPriority>NotAMapViewListener</ModelessCommandPriority>
   <ModelessListenerMask>0</ModelessListenerMask>
</ApplicationCommand>
```

3.2. Beispiel einer ini-Konfigurationsdatei für Usercommands

Der Inhalt einer ini-Konfigurationsdatei für die Registrierung als Usercommand sieht exemplarisch folgendermassen aus:

```
[ModalCmd]
ProgID=MyGMCommands.ModalCmd
CommandType=0
EnableMask=1
ListenerMask=0
IsModal=1
Description=Beschreibung der Funktionalität des Customcommands
ToolTip=Kurzbeschreibung Customcommand
SmallMonoBitmapPath=
SmallColorBitmapPath=
LargeMonoBitmapPath=
LargeColorBitmapPath=
```

3.3. Parameterwerte für `EnableMask`

Der Parameter `EnableMask` legt die Aktivierungsbedingungen für den GeoMedia Commandserver fest. Die Angaben zu den möglichen Parameterwerten fehlen in der GeoMedia-Dokumentation. Es gelten die in Tabelle 1 angeführten Werte. Mehrere (Standard-)Aktivierungsbedingungen (`EnableMask` ungleich `1`) können durch Addieren kombiniert werden. Sollen

[8] Toolbar-Bitmaps können bei der Registrierung als Usercommand unter .NET nicht mehr eingesetzt werden. Die Bitmap-Angaben in der ini-Konfigurationsdatei werden in jedem Fall ignoriert und können demzufolge leer bleiben.

beispielsweise die Aktivierungsbedingungen *Map view is active* und *Select set count is one* gelten, ist für `EnableMask` der Wert `327680` (`65536`+`262144`) einzusetzen.

Aktivierungsbedingung	**Parameterwert**
No enabling conditions	`0`
Custom enabling requirements (Der GeoMedia Commandserver ruft die Methode `CanEnable` auf.)	`1`
Data view is active	`2`
Data view one column selected	`4`
Data view columns selected	`8`
Data view one row selected	`16`
Data view rows selected	`32`
Data view one cell selected	`64`
Data view cells selected	`128`
Data view column type is number	`256`
Data view column type is hypertext	`512`
Data view row being edited	`1024`
Geoworkspace is open	`2048`
One or more queries exist	`4096`
One or more connections exist	`8192`
Readable connections exist	`49152`
Writable connections exist	`49152`
Map view is active	`65536`
Data view or map view is active	`131072`
Select set count is one	`262144`
Select set count is one or more	`524288`
Select set has geometry feature	`1048576`
Select set has raster	`2097152`
Select set has records	`4194304`
Select set has non-raster	`8388608`
Select set count is one or more OR Data view has cells or columns selected	`33554432`
Data view has rows	`67108864`
Data view has columns	`134217728`
Data view is updateable	`268435456`
Two or more connections exist	`536870912`
Legend is visible	`1073741824`

Tabelle 1: Parameterwerte für `EnableMask`