Projekt Exquisite

Kurzdokumentation

März 2013

Inhaltsverzeichnis

Vorwort

Hinweise zur vorliegenden Kurzdokumentation:

Datum der letzten Bearbeitung: 11.03.2013

zuletzt bearbeitet durch: Arash Baharloo

Dieses Handbuch ist einer laufenden Bearbeitung unterworfen

Copyright / Lizenz:

Teil I:
Einführung in Exquisite

1 Einleitung

1.1 Hintergrund

1.2 Ziele

2 Einführung in die Constraint-Programmierung

2.1 Constraints, Constraint-Systeme und Constraint-Solver

2.2 Choco Constraint-Solver

3 Einführung in die Excel 2010-Programmierung mit VSTO und .NET 4.0

Es werden immer noch die meisten auf Microsoft Office basierende Anwendungen ohne Visual Studio Tools für Office (VSTO) umgesetzt. Dabei bietet VSTO die Möglichkeit, Softwarelösungen für Office Anwendungen deutlich schneller, eleganter und bei manchen Aufgaben sogar effizienter zu entwickeln.

VSTO bietet nicht nur einen einfachen Einstieg in die Office Programmierung, sondern auch die volle Unterstützung des .NET Frameworks.

3.1 Überblick VSTO

Visual Studio Tools für Office (VSTO) ist im Wesentlichen eine Menge von Projektvorlagen in und für die Entwicklungsumgebung Visual Studio, die von Microsoft entwickelt und bereitgestellt wird. Mit diesen Vorlagen ist es möglich, Lösungen für Office Anwendungen unter Verwendung der .NET Programmiersprachen (Visual Basic oder Visual C#) zu entwickeln. Da so entwickelte Lösungen auf .NET Framework basieren, verbinden sie die Office Anwendungen mit .NET Framework im vollen Umfang.

3.2 Das Objektmodell von Excel

Die Objektmodelle der Office Anwendungen basieren auf COM. Um aus einer .NET Anwendung auf diese COM-Komponenten zugreifen zu können, werden besondere Wrapperklassen, s.g. Interop-Assemblies, benötigt. Wenn ein neues Office-Projekt in Visual Studio erstellt wird, fügt Visual Studio automatisch Verweise auf die primären Interopassemblys (PIAs) von Microsoft Office hinzu, die zum Erstellen des Projekts erforderlich sind. Die PIAs werden in vorkompilierter Form von Microsoft verteilt.

Die PIAs beinhalten eine Menge von signierten Wrapperklassen und ermöglichen verwaltetem Code, mit dem COM-basierten Objektmodell einer Microsoft Office-Anwendung zu interagieren.[1]

Das Verständnis, wie Excel intern arbeitet, ist eine besonders wichtige Grundlage für die Programmierung einer Excel-basierte Anwendung. Abbildung 3-1 Zeigt einen Ausschnitt aus dem Objektmodell von Excel. Mehr Informationen zum Excel-Objektmodell finden Sie in der offiziellen Dokumentation [1].

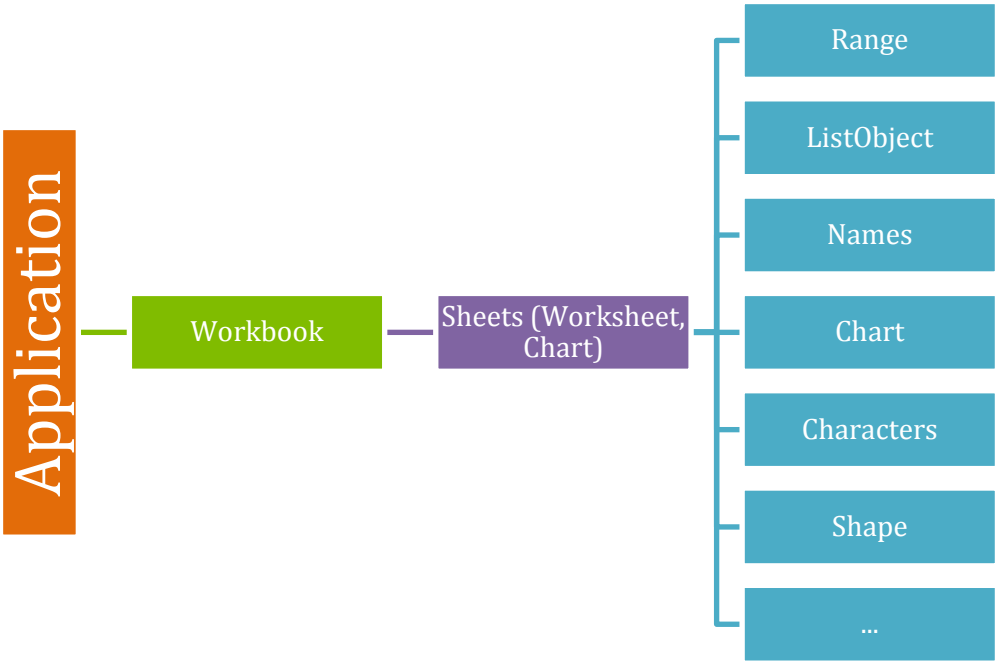

Abbildung 3-1: Ausschnitt aus dem Excel-Objektmodell

Hinweis: Es existiert in Excel keine Cell-Objekte für die einzelnen Zellen in einem Arbeitsblatt, sondern Range-Objekte, die sowohl eine einzelne Zelle als auch einen Bereich von mehreren Zellen repräsentieren.

3.3 Interaktion mit anderen Technologien

[1] Mehr zu PIA: http://msdn.microsoft.com/de-de/library/15s06t57%28v=vs.100%29.aspx

3.3.1 Das neue Dateiformat von Office (Excel) im Überblick

3.3.2 Custom XML Parts

4 Einführung in Exquisite

4.1 Das Excel-Add-In (Client)

4.2 Der Exquisite-Server

4.3 Die Datenhaltung und Kommunikation

Teil II:
Exquisite Systemstruktur

5 Konzeption und Vorüberlegungen

5.1 Das Client-Server-Modell

5.2 XML als Schnittstelle und neutrales Sicherungsformat

5.3 Choco als Constraint-Solver

6 Der Client

Ein Excel Add-In auf Anwendungsebene bildet den Client des verteilten Programms. In diesem Kapitel werden zunächst die Systemanforderungen und Voraussetzungen beschrieben. Dies soll aufzeigen, welche Systembestandteile zwingend notwendig sind, um eine prinzipielle Lauffähigkeit der Exquisite-Software zu erreichen. Anschließend werden die Werkzeuge, die Installationsschritte und die Konfigurationsmöglichkeiten genauer erläutert. Die Angaben zur empfohlenen Konfiguration beruht auf Erfahrungswerten aus dem Praxiseinsatz.

6.1 Systemanforderungen und Voraussetzungen

Das System wurde in der folgenden Umgebung entwickelt und getestet:

- Windows 7 x64 mit Service Pack 1
- Office (Excel) 2010 x86
- Visual Studio 2010 mit Service Pack 1

Höchstwahrscheinlich wird die Verwendung höherer Versionen von Windows, Visual Studio und Office problemlos möglich sein. Generell lassen sich aber über andere Versionen keine genaue Aussagen treffen, da wir das Programm auf anderen Systemen nicht getestet haben.

Der Einsatz einer Java-Laufzeitumgebung ist für den Client nicht zwingend erforderlich. Für die Nutzung des kompletten Funktionsumfangs (Diagnose) wird er aber dringend benötigt. Weitere Informationen zu diesem Thema finden Sie unter anderem im Abschnitt 7.1.

6.2 Werkzeuge

Für die reibungslose Weiterentwicklung des Programms werden folgende Werkzeuge benötigt:

- Visual Studio 2010 Professional oder höher (mit Service Pack 1)
- AnkhSVN[2] – Subversion Unterstützung für Visual Studio
- Office (Excel) 2010
- Optional: R# ReSharper[3] – ReSharper wertet Visual-Studio um diverse Funktionen auf, die vor allem beim Entwickeln sehr hilfreich sind

Wir haben uns für die Verwaltung des Quellcodes in Visual Studio für AnkhSVN entschieden, da es kostenlos ist, vollständig in Visual Studio integriert ist und einige interessante Features bietet.

Hinweis: Die kostenlosen Versionen von Visual Studio (die Express Editionen) bieten keine Unterstützung für VSTO

Hinweis: Es können mit Visual Studio nur Lösungen für Office-Versionen entwickelt werden, die auch lokal auf dem Entwicklungsrechner installiert ist. Eine parallele Installation von verschiedenen Office Lösungen wird von VSTO nicht unterstützt.

6.3 Die Installation

6.3.1 Installationsdateien überprüfen

Es empfiehlt sich immer, die Installationsdateien auf Richtigkeit und Vollständigkeit zu überprüfen.

Hinweis: Visual Studio 2010 Ultimate ist auf dem LS13-Server unter *„Technical stuff\Software\Visual Studio 2010 Ultimate“* zu finden (Deutsch und English).

Hinweis: Office Professional Plus 2010 finden Sie auf dem LS13-Server unter *„Technical stuff\Software\Office Software\Office Professional Plus 2010 update“*

Hinweis: Die neusten Versionen von AnkhSVN und ReSharper finden Sie unter den jeweiligen Webseiten (siehe Abschnitt 6.2).

[2] http://ankhsvn.open.collab.net/
[3] http://www.jetbrains.com/resharper/

6.3.2 Der Installationsverlauf

Prinzipiell sollten jeweils alle für die verwendeten Windows Version verfügbaren Service Packs installiert sein.

Falls Sie es noch nicht getan haben, melden Sie sich für die Installation bitte als Administrator auf Ihrem System an.

Visual Studio 2010 installieren

Ist bereits eine (ältere) Version von Visual Studio installiert, beenden Sie alle zugehörigen Programme. Es empfiehlt sich die bereits installierte Version von Visual Studio vor der Installation von Visual Studio 2010 vollständig zu deinstallieren. Darüber hinaus sollten Sie lieber vorhandene Virenscanner für die Dauer der Installation deaktivieren.

Hinweis: eine offizielle Anleitung zur Installation von Visual Studio finden Sie hier: *http://msdn.microsoft.com/de-de/library/vstudio/e2h7fzkw%28v=vs.100%29.aspx*

Office (Excel) 2010 installieren

Ist bereits eine (ältere) Version von Microsoft Office installiert, beenden und deinstallieren Sie alle zugehörigen Programme. Office 2010 lässt sich am leichtesten installieren, wenn es auf dem System vorher noch keine Office Version installiert war.

Hinweis: eine Schritt für Schritt Anleitungen zur Installation von Office 2010 finden Sie unter: *http://www.office2010-hilfe.de/knowledge/kb_show.php?id=51*

AnkhSVN installieren

Beenden Sie Visual Studio und alle zugehörigen Programme vor der Installation von AnkhSVN und starten Sie die Installation mittels Doppelklick auf die AnkhSVN Setup-Datei. Starten Sie anschließend Visual Studio (neu).

Hinweis: weitere Anleitungen und Informationen zu AnkhSVN finden Sie unter: *http://help.collab.net/index.jsp?topic=/com.collabnet.doc.anksvn_001/action/ankh_getting_started.html*

6.4 Konfiguration

In den folgenden Abschnitten werden die Konfiguration der Installierten Anwendungen und Einrichtung des Exquisite-Projektes Schritt für Schritt zum Nachvollziehen bzw. zum Mitmachen beschrieben:

SVN-Repository einrichten

Gehen Sie wie folgt vor, um in Visual Studio mittels AnkhSVN ein neues Repository einzurichten:

1. Legen Sie in Visual Studio zuerst ein neues Repository an. Dazu klicken Sie auf Ansicht und dann auf Repository Explorer (siehe Abbildung 6-1).

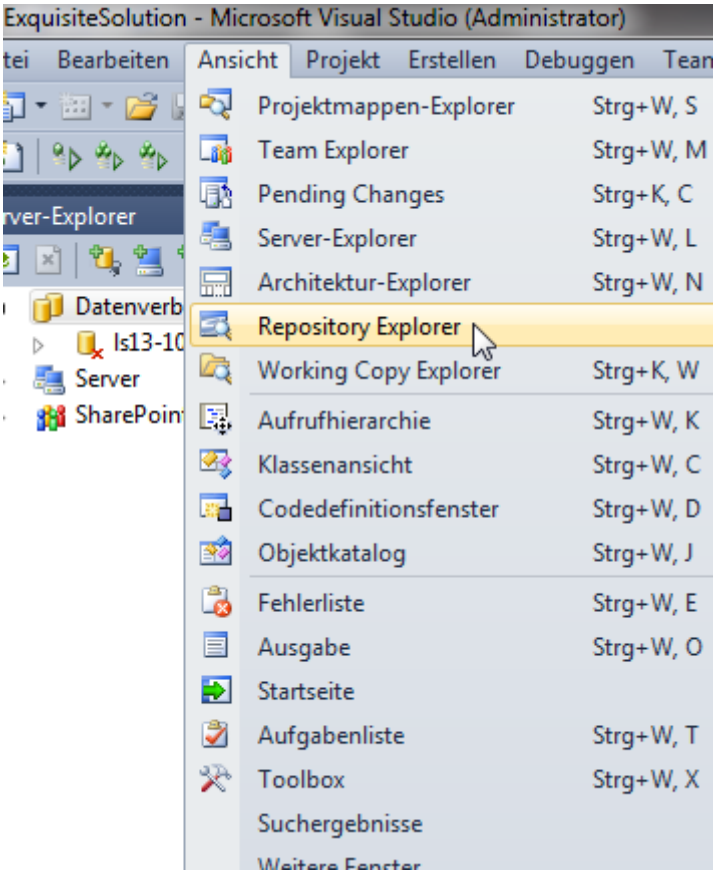

Abbildung 6-1: Repository Explorer aufrufen

2. Fügen Sie eine neue SVN-URL in den Repository Explorer (siehe Abbildung 6-2).

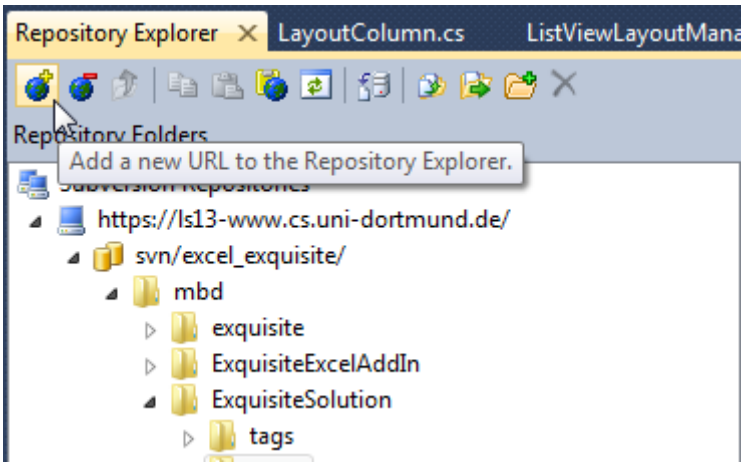

Abbildung 6-2: Eine neue URL hinzufügen

Projekt auschecken

Gehen Sie wie folgt vor, um die Exquisite-Projektmappe aus dem LS13-SVN-Server in Visual Studio einzuchecken.

1. Geben Sie *„https://ls13-www.cs.uni-dortmund.de/svn/excel_exquisite/“* ein und bestätigen Sie Ihre Angaben mit OK.
2. Geben Sie bei Bedarf Ihre SVN-Zugangsdaten ein.
3. Es ist nun möglich das Repository zu durchsuchen. Wählen Sie den Ordner „/excel_exquisite/mbd/ExquisiteSolution/trunk/“ und öffnen Sie die Projektmappe „ExquisiteSolution.sln“ mit einem Doppelklick (siehe Abbildung 6-3).

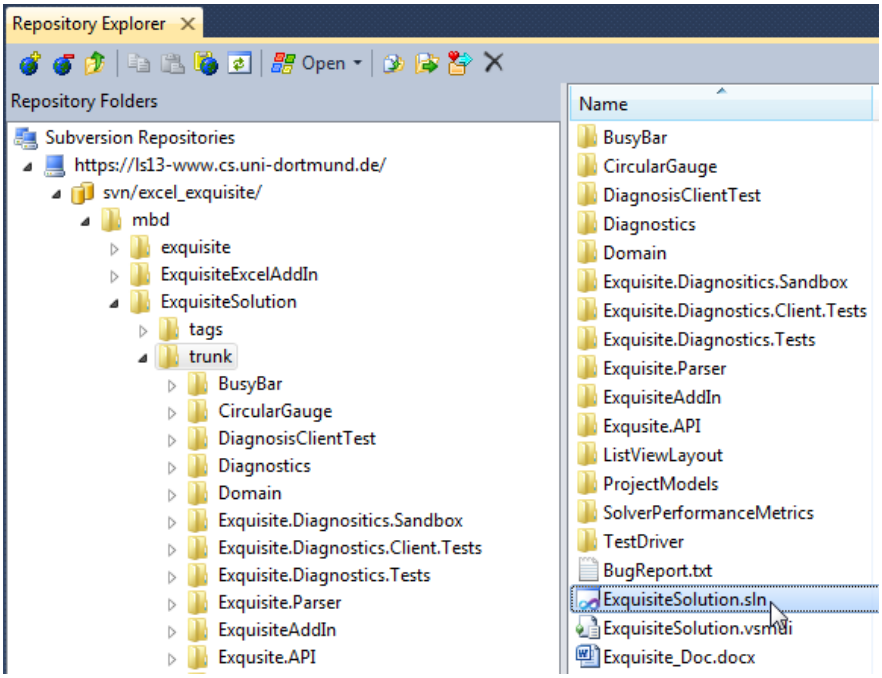

Abbildung 6-3: Projektmappe auswählen

4. Die Exquisite Projektmappe und alle dazugehörige Projekte sind nun in Visual Studio verfügbar und können bearbeitet und ausgeführt werden.

Hinweis: Die SVN-Befehle befinden sich im Kontextmenü zum Projekt oder den Projektdateien. Wichtige Befehle:

- *Commit:* Hiermit werden aktuelle Veränderungen in das Repository hochgeladen.
- *Revert:* Hiermit werden aktuelle Veränderungen in Visual Studio zurückgesetzt.
- *Update:* Hiermit werden die Dateien auf die aktuelle Version (Head) oder eine ausgewählte Version synchronisiert.

Hinweis: Die Schaltflächen „*Revert*“ und „*Commit*“ erscheinen nur bei Veränderungen.

Startprojekt festlegen

Um das Excel-Add-In aus Visual Studio heraus starten zu können, muss das entsprechende Projekt als Startprojekt festgelegt sein. Wählen Sie dazu das Projekt „Exquisite.ExcelAddIn“ mit einem Rechtklick aus und klicken sie den Befehl „Als Startprojekt festlegen“ (siehe Abbildung 6-4).

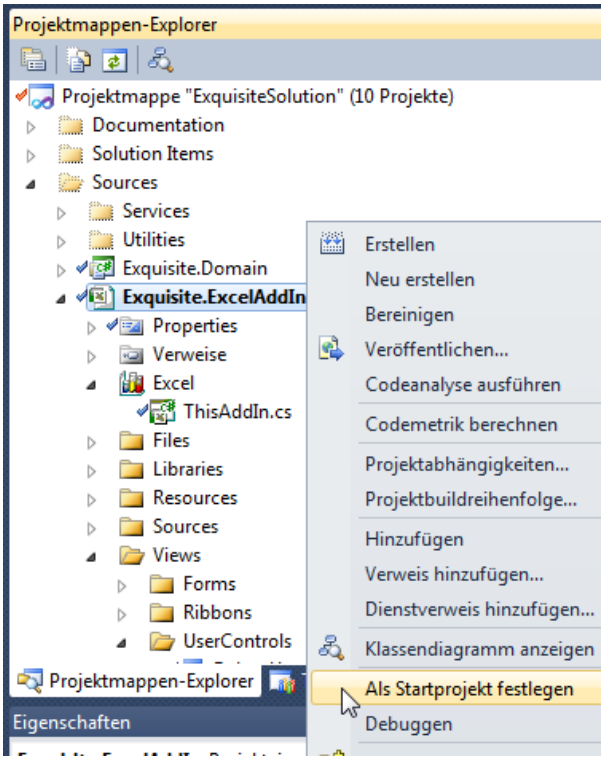

Abbildung 6-4: Startprojekt festlegen

Projekt ausführen

Wenn das Projekt nun aus Visual Studio heraus gestartet wird, wird die dazugehörige Office-Anwendung (Excel) geöffnet und das Add-In in der Anwendung geladen. Beim Beenden der Excel-Anwendung befindet man sich als Entwickler wieder in der Oberfläche von Visual Studio.

Hinweis: Das Add-In ist weiterhin in Excel verfügbar und wird automatisch beim Start von Excel geladen. Ein Add-In auf Anwendungsebene muss also manuell aus der Office-Anwendung entfernt werden, sofern es nicht weiter benötigt wird.

Mehr zum Add-In finden Sie im Abschnitt 8.

7 Der Server

7.1 Systemanforderungen und Voraussetzungen

7.2 Werkzeuge

7.3 Installation

7.3.1 Installationsdateien überprüfen

7.3.2 Der Installationsverlauf

7.4 Konfiguration

Teil III:
Exquisite als integriertes Werkzeug zur Fehlersuche

8 Der erste Programmstart

8.1 Der Debug-Mode

9 Die Benutzerschnittstelle

9.1 Exquisite-Menüband (Ribbon)

9.1.1 Testfälle

9.1.2 Diagnose

9.1.3 Konfigurationsmöglichkeiten

9.1.4 Werkzeuge (Utilities)

9.2 Exquisite-Steuerelemente (UserControls)

9.2.1 Testfallmodellierung

9.2.2 Debug-Fenster und visuelle Feedbacks

10 Diagnose

Diagnose Starten

Ergebnisse der Diagnose

Teil IV:
Schlussbemerkungen

11 Häufige Fehler

12 Häufig gestellte Fragen

Anhang

A. Entwicklungsumgebung

Die Entwicklungsumgebung besteht aus folgenden Komponenten:

Alle lizenzfreien Komponenten befinden sich auf dem LS13-Server.

B. Verzeichnisstruktur des Servers

Verzeichnis	Beschreibung

C. Verzeichnisstruktur des Clients

Verzeichnis	Beschreibung

Glossar

Schlüsselwort	Bedeutung

Literaturverzeichnis

[1] Microsoft, „Excel Object Model Reference,“ 2011. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff846392.aspx. [Zugriff am 06 03 2013].

Abbildungsverzeichnis

Index