

**.NET & More
Visual Studio**

dot.net magazin

.NET, Visual Studio & More

AUF CD:

- Advantage Database Server 9.0
- Telerik FormDecorator 2008
- (smartassembly) 3.2
- Shell MegaPacks 2009
- SQL Server 2005 Performance Dashboard Reports
 - Licence Protector
 - PostSharp 1.0.11.425
 - code2plan
 - RTF Converter
 - Mono 2.2
 - IKVM.NET
 - u.v.m.



DOT.NET TV

Showcase: Start-up mit Silverlight

SharePoint typischer programmieren



Mobility 2.0?

Innovative mobile Anwendungen

► 117

Datenträger enthält
Info- und
Lehrprogramme
gemäß §14 JuSchG

Step by Step reloaded

Das MVC-Framework im Einsatz ► 80



**Sonderdruck
der Firma Fecher**

Controller

Den Überblick behalten

Aspektorientierte Programmierung mit dem ASP.NET-MVC-Framework ► 76

Agile Methoden

Die Crystal-Methoden-Familie ► 111

C# auf Knopfdruck

Aus einer Java-Klasse einfach eine C#-Klasse migrieren

Hier soll nicht schon wieder Java mit C# verglichen werden oder etwa die Zukunftsaussichten der Technologien ausgelotet. Es wird vielmehr gezeigt, wie man mit dem Java2CSharpTranslator Java in C# übersetzt.

von Daniel Basler

Es ist für Entwickler immer wieder schwer, sich von Vertrautem zu trennen. Es kann aber sinnvoll sein, seine in Java entwickelte Applikation oder sein Framework auch für die Sprache C# anzubieten. Hierbei kann sehr wohl die Migration im Vordergrund stehen, da sich über Jahre hinweg entwickelte Klassen nicht eben mal so migrieren lassen, stellt dies das Management und die IT-Abteilung vor große Herausforderungen.

Neubau oder Migration?

Die Frage, ob eine Applikation auf Basis der neuen Plattform .NET komplett neu

entwickelt werden, oder ob Klassen automatisch migriert werden sollten, lässt sich leider nicht so einfach ad hoc beantworten. Hierfür gelten meistens folgende Kriterien:

- **Kosten:** Eine komplette Neuentwicklung ist um einiges teurer als eine Migration.
- **Zeitfaktor:** Die Entwicklung und Implementierung einer neuen Software dauert erheblich länger als die Anwendung eines Migrations-Tools.
- **Risiko:** Neben Kosten- und Zeitrisko sollten Sie Design- und Codierungsfehler miteinberechnen, da diese auch in Ihrem neuen Projekt auftreten.

Das Werkzeug Java2CsharpTranslator, das Open Source verfügbar ist, ermöglicht es, die oben genannten Faktoren und den damit verbundenen Migrationsprozess überschaubar zu gestalten.

Das Werkzeug von ILOG steht unter [1] zur Verfügung. Hier benötigen Sie die beiden jar-Dateien *com.ilog.rules.Java2CsharpTranslator_1.x.jar* und *org.eclipse.core.filebuffers_3.3.1.r331_x.jar* sowie natürlich als Entwicklungsumgebung Eclipse ab Version 3.x. Die Grund-

idee hierbei ist, dass Java und C# sehr viele gemeinsame Konzepte besitzen. Eclipse wurde als Werkzeug gewählt, weil es über einen sehr guten Plug-in-Mechanismus sowie Änderungs- und Refactoring-Fähigkeiten verfügt. Der Plug-in-Mechanismus macht sich schon bei der Installation des Translators bemerkbar. Legen Sie beide jar-Dateien einfach in das Plug-in-Verzeichnis von Eclipse ab. Starten Sie danach Eclipse, so steht Ihnen das Plug-in direkt zur Verfügung.

Schritte der Migration

Ein Projekt zur Migration gliedert sich in vielerlei Hinsicht genauso wie ein gewöhnliches Softwareprojekt mit seinen einzelnen Phasen. Das heißt, in der ersten Phase sollte die Bestandsaufnahme der vorhandenen Klassen/Applikationen erfolgen. Sind die ersten Projektphasen abgeschlossen, kann die Codegenerierung, die hier im Beispiel auch gezeigt werden soll, beginnen. Nach der Installation des Plug-ins starten Sie Eclipse mit den zu migrierenden Projektdateien.

Ist das Projekt in Eclipse geladen, so kann man jetzt über FILE | EXPORT | OTHER | ILOG JAVA TO CSHARP TRANSLATOR den

Inhalt

Die Migration von Java nach C#

Zusammenfassung

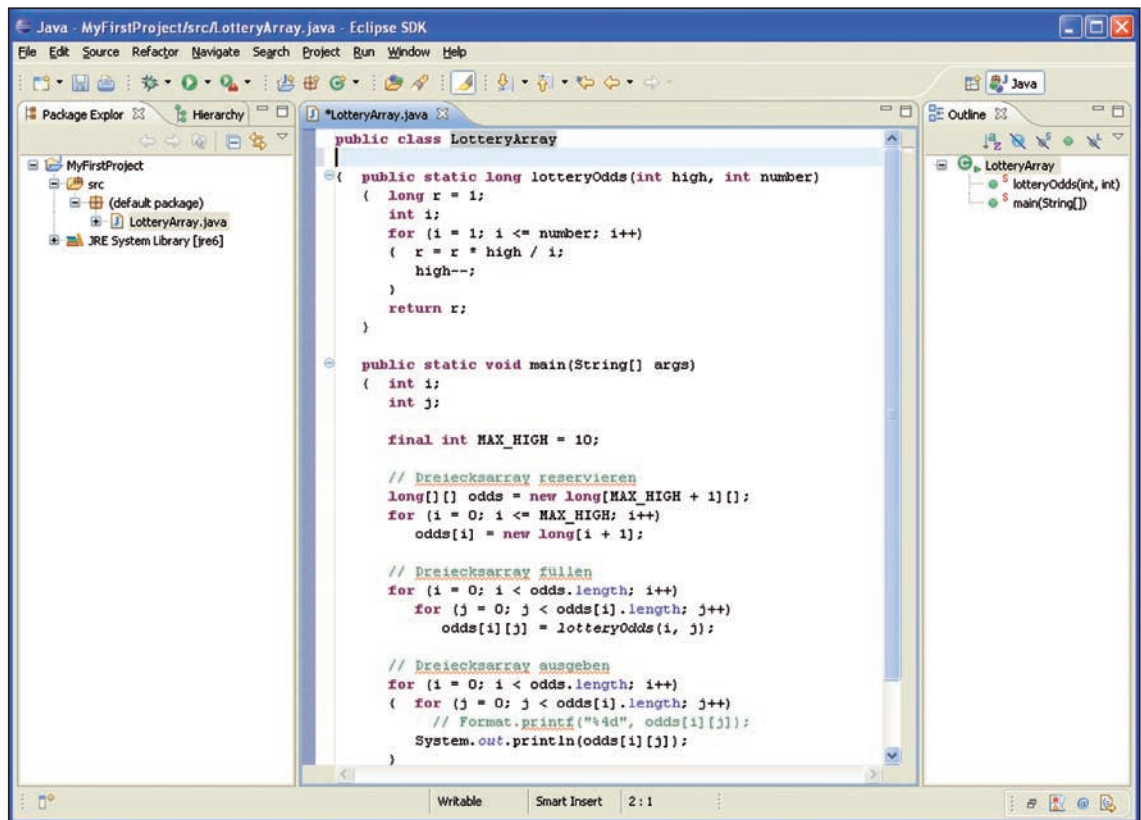
Das Unternehmen ILOG hat seinen Java Translator Open Source zur Verfügung gestellt. Es wird hier die Migration von Java nach C# geprüft

Quellcode

Java und C#

kurz & bündig

Abb 1. Das Migrationsprojekt in Eclipse



Migrationsprozess starten. Selektieren Sie das Projekt in der Listenauswahl und klicken Sie auf den Button NEXT. In dem Dialogfenster TRANSLATOR OPTIONS können Sie noch entsprechende Optionen einstellen, so zum Beispiel das Verzeichnis für die C#-Dateien. Über den Button NEXT können Sie weitere Einstellungen vornehmen. Für das Beispiel wird die Übersetzung mit dem Button FINISH gestartet.

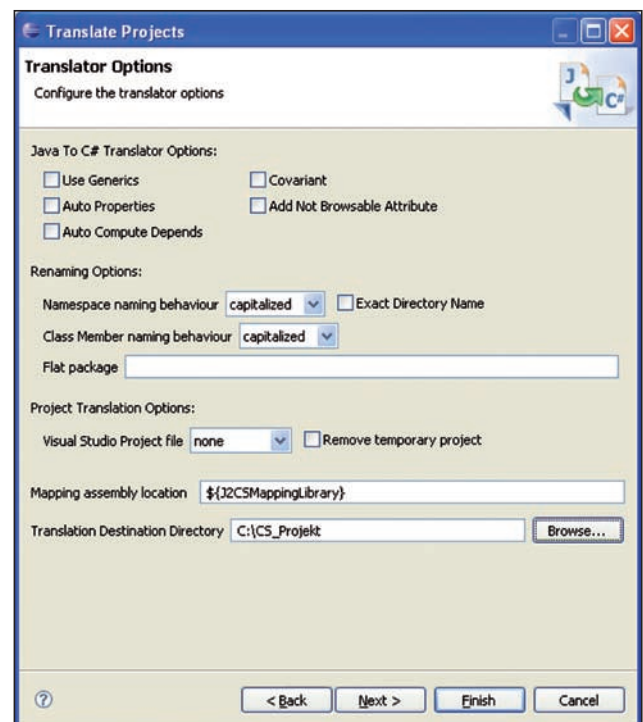
In Eclipse wird jetzt ein weiteres Projekt mit den übersetzten Dateien angelegt. Die vom Migrations-Tool erzeugten C#-Dateien befinden sich unter dem im Options-Dialogfenster angegebenen Ordner. Hier können jetzt ein entsprechendes Visual-Studio-Projekt eingerichtet und die Klasse eingebunden werden. Für bestimmte Mapping-Funktionalitäten müssen Sie in Ihr C#-Projekt die Datei *Ilog.J2CsMapping.dll* einbinden. Auch diese steht Ihnen unter Sourceforge zur Verfügung.

Abbildung 3 zeigt die übersetzte Java-Klasse in einem Visual-Studio-Projekt. Diese einfache Klasse ist unter C# auch sofort lauffähig.

Das Ziel

Ziel der Migration ist ein kompilierbares .NET-Projekt. Hierbei wird bei der Migration automatisch eine Umsetzung der Java-Konstrukte in entsprechende .Net-Konstrukte und Strukturen vorge-

Abb 2. Translator Options



nommen. Das Ergebnis der Portierung ist also reiner C#-Code, auch in Hinblick auf Events, native Datentypen und die Ablaufumgebung. Komplett übersetzbarer und lauffähiger Code entsteht erfahrungsgemäß erst durch manuelle Nacharbeit oder durch eine Umstellung des Java-Codes im Vorfeld. Der Translator von ILOG unterstützt schon sehr gut die automati-

sche Migration von Java nach C#, trotz alledem sind immer noch viele Workarounds notwendig, um Java nach C# zu übersetzen. So werden zum Beispiel keine Methodenaufrufe in inneren Klassen unterstützt. Sehr häufig müssen Methoden per *Rename* umbenannt werden. Auch die Übersetzung von Subklassen und Collections gestaltet sich schwierig. Hier sollte

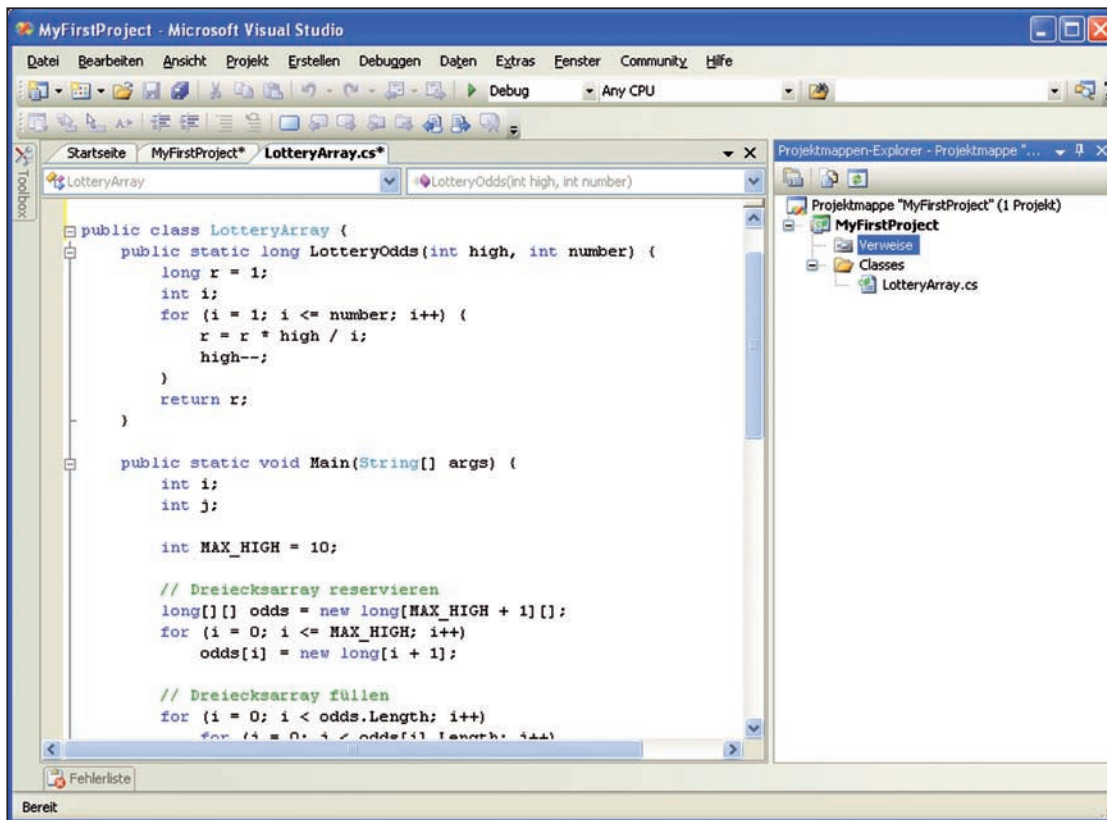


Abb 3. Die migrierte Klasse

im Vorfeld der Java-Code entsprechend angepasst werden. Ziel der Übersetzung ist, das kompilierbare Projekt in eine Applikation oder ein Framework zu verwandeln, die bzw. das lauffähig ist und auch genau das Verhalten der Ursprungsapplikation/des Frameworks aufweist. Hierfür benötigt man aber, wie auch schon im Vorfeld, großes Know-how und Erfahrungen in beiden Sprachwelten.

In der erzeugten *Translation.xml*-Datei finden Sie Warnmeldungen zu möglicherweise nicht adäquat umsetzbaren Java-Konstrukten. Auch hier müssen diese Einträge dann manuell realisiert werden. Besitzt man dann einen entsprechenden C#-Code, stehen erste funktionale Tests der Software und die sich daraus ergebenden Maßnahmen zur Problembeseitigung an.

Aufgrund der Komplexität der Konvertierung können nicht alle Java-Konstrukte konvertiert werden. Der ILOG Translator gibt in der *Translation.xml*-Datei die Aufgaben aus, die im Code nicht konvertiert werden konnten. Diese Anpassungen müssen zum Abschluss der Konvertierung durchgeführt werden. Bedenken Sie hierbei immer, dass die Java-API-Aufrufe in systemeigene .NET-Framework-Aufrufe konvertiert werden. Die Entwickler des Java2CSharpTranslators haben viel Wert auf die möglichst komplette Automatisierung der Java-Konstrukte gelegt. Dies

bedeutet aber, dass neue Konstrukte unter .NET während des Migrationsprozesses eingebracht werden. Daher ist auch eine hundertprozentige Unterstützung nicht möglich. Das stellt einige spezielle Herausforderungen an das Migrationsteam.

Unsupported Patterns

So bleibt ein überschaubarer Rest an Dingen, die sich nicht einfach auf Knopfdruck von Java nach C# umsetzen lassen. Hier ist im Vorfeld oder Nachhinein der manuelle Eingriff des Entwicklers gefragt. So müssen konsequent alle Sprachelemente, die sich nicht vollautomatisch umsetzen lassen, nachbearbeitet werden. Hier ist hin und wieder auch die eigene Implementierung unter .NET einfacher zu bearbeiten als unter Java. Von einer Migration von Swing oder SWT sollte man absehen. Hier ist es sinnvoller, das GUI direkt unter C# zu entwickeln. Hier kommt es sehr häufig zu sehr großen Unstimmigkeiten wegen der unterschiedlichen Implementierung von Steuerelementen.

Fazit

Mit dem Java Translator von ILOG steht ein effizienter Fundus an Software und Know-how zur Verfügung. Es lohnt sich vor allem für umfangreiche Frameworks, diese mit einem kalkulierbaren Aufwand nach C# zu migrieren. Dadurch, dass der

Übersetzer Open Source zur Verfügung steht, befreit er das Migrationsteam erst einmal von kostentreibenden Tools, beziehungsweise einer kompletten Neuentwicklung. Darüber hinaus stehen der migrierten Anwendung alle Vorteile von .NET zur Verfügung. Achtet man im Vorfeld auf seine Java-Konstrukte, so hält sich auch die manuelle Nacharbeit in einem überschaubaren Rahmen. Weiterhin sind auch Anpassungsmöglichkeiten Open Source vorhanden.



Daniel Basler ist Senior Consultant bei der fecher e. Kfm und innerhalb der Abteilung Services für die Portierung von SAL-Applikationen nach .NET verantwortlich.

Links & Literatur

- 1 <http://sourceforge.net/projects/j2ctranslator>

fecher. e. Kfm.

Seestraße 2-4
63110 Rodgau
Telefon: (06106) 605-0
Fax: (06106) 605-200
www.fecher.eu