

Testdaten automatisch generieren

# Meier, Müller, Schulze ...



Nach wie vor spielt die klassische „Forms over Data“-Anwendung eine große Rolle. Daten aus einer Datenbank sollen per Formular bearbeitet werden. Wenn diese Applikationen getestet werden, spielen Testdaten eine zentrale Rolle. Möglichst viele sollten es sein und möglichst realistisch geformt noch dazu. Stefan, fällt dir dazu eine Übung ein?

dnpCode: A1005dojo

Wer übt, gewinnt

In jeder dotnetpro finden Sie eine Übungsaufgabe von Stefan Lieser, die in maximal drei Stunden zu lösen sein sollte. Wer die Zeit investiert, gewinnt in jedem Fall – wenn auch keine materiellen Dinge, so doch Erfahrung und Wissen.

Es gilt:

- Falsche Lösungen gibt es nicht. Es gibt möglicherweise elegantere, kürzere oder schnellere Lösungen, aber keine falschen.
- Wichtig ist, dass Sie reflektieren, was Sie gemacht haben. Das können Sie, indem Sie Ihre Lösung mit der vergleichen, die Sie eine Ausgabe später in dotnetpro finden.

Übung macht den Meister. Also – los geht's. Aber Sie wollten doch nicht etwa sofort Visual Studio starten ...



Immer wieder begegnet man der Anforderung, Daten aus einer Datenbank in einem Formular zu visualisieren. Oft sind die Datenmengen dabei so groß, dass man nicht einfach alle Daten in einem Rutsch laden sollte. Stattdessen müssen die Daten seitenweise abgerufen und visualisiert werden. Suchen und Filtern kommen meistens hinzu, und schon stellt sich die Frage, ob der gewählte Ansatz auch noch funktioniert, wenn mehr als nur eine Handvoll Testdaten in der Datenbank liegen.

Solche Tests auf Echtdateien Ihrer Kunden vorzunehmen wäre übrigens keine gute Idee. Diese unterliegen dem Datenschutz und sollten keinesfalls zu Testzwecken verwendet werden. Und für eine völlig neue Anwendung stehen natürlich noch gar keine Echtdateien zur Verfügung. Folglich bleibt nur die Möglichkeit, Testdaten zu generieren. Und genau darum geht es in dieser Übung: Erstellen Sie eine Bibliothek zum Erzeugen von Testdaten.

## Verschiedene Arten von Testdaten

Die generierten Testdaten sollen eine Tabellenstruktur haben. Für jede Spalte wird definiert, von welchem Typ die Werte sind und wie sie erzeugt werden. Anschließend gibt man an, wie viele Zeilen generiert werden sollen, und die Testdaten werden generiert.

Die Anforderungen an die Daten können sehr vielfältig sein. Um hier ausreichend flexibel zu sein, sollen die Daten nach verschiedenen Strategien erzeugt werden können. Reine Zufallsdaten sind ein erster Schritt, dürften aber in vielen Fällen nicht

ausreichen. Zumindest eine Beschränkung innerhalb vorgegebener Minimum- und Maximumwerte erscheint sinnvoll.

Eine weitere Strategie könnte darin bestehen, eine Liste von möglichen Werten vorzugeben, aus denen dann zufällig ausgewählt wird. So könnten beispielsweise Straßennamen generiert werden, die in den Formularen dann auch wie Straßennamen aussehen

statt wie zufällig zusammengewürfelte Zeichenfolgen. Es müssen lediglich einige Straßennamen vorgegeben werden. Das Gleiche bietet sich für die Namen von Personen an. Auch hier kann gut mit einer Liste von Namen gearbeitet werden, aus der dann zufällig Werte ausgewählt werden.

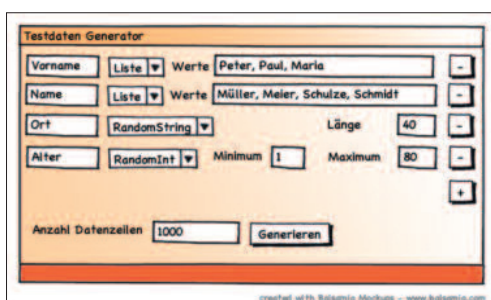
Die Strategie für die Testdatenerzeugung soll möglichst flexibel sein. Ein Entwickler sollte mit wenig Aufwand einen eigenen Generator ergänzen können. Endergebnis der Datenerzeugung soll eine Aufzählung von Zeilen sein:

```
IEnumerable<object[]>
```

Die generierten Zeilen können dann beliebig verwendet werden. Sie können direkt in Tests einfließen oder auch zuerst als Datei gespeichert werden. Hier bietet sich beispielsweise die Speicherung als CSV-Datei an. Auch das Speichern in einer Datenbank ist natürlich ein typisches Szenario. Das konkrete Speichern der Daten sollte unabhängig sein vom Erzeugen. Es lohnt sich also wieder, sich vor der Implementierung ein paar Gedanken zur Architektur zu machen.

Auch bei dieser Übung geht es wieder primär um eine Bibliothek und weniger um eine Benutzerschnittstelle. Wer mag, kann sich aber auch um eine Benutzerschnittstelle kümmern, denn die dürfte hier etwas anspruchsvoller sein. Schließlich benötigen die verschiedenen Generatoren unterschiedliche Eingabedaten. Genügen bei einem Zufallsgenerator vielleicht Minimum und Maximum, müssen bei einem anderen Generator Wertelisten eingegeben werden. Hinzu kommt, dass die Eingabedaten von unterschiedlichem Typ sein können, wofür unterschiedliche Eingabevalidierungen nötig sind. Abbildung 1 zeigt eine erste Skizze einer Benutzerschnittstelle.

Und denken Sie stets an die Musiker: Die verbringen die meiste Zeit mit Üben, nicht mit Auftritten! Wir Softwareentwickler sollten auch regelmäßig üben, statt immer nur zu performen. Schließlich sollte man beim Auftritt keine Fehler machen, nur beim Üben ist das zulässig und sogar erwünscht: ohne Fehler keine Weiterentwicklung. Also üben Sie und machen Sie Fehler! [ml]



[Abb. 1] So könnte das GUI für einen Testdatengenerator aussehen.