OPTIMALE WEBUMGEBUNG MIT ANGULARJS UND ASP.NET, TEIL 2

Gute Kommunikation

Entwurfsmuster und Frameworks zum Einbinden von AngularJS in .NET-Anwendungen.

In einer typischen ASP.NET-MVC- oder Web-Forms-Anwendung kann es leicht geschehen, dass eine saubere Trennung von Daten und Layout verloren geht. Bei einer Single-Page-Anwendung (SPA) besteht die Möglichkeit, den Datenfluss zu überdenken und neu zu definieren. Dabei bietet sich eine REST-Architektur an [1]. Doch hinsichtlich der Protokolle, Formate und Konventionen bleiben Fragen für die praktische Umsetzung von REST offen: Wie sollen etwa die Abfrageparameter heißen? Welchem Format soll eine Antwort genügen? Wie lassen sich die Schnittstellen maschinenlesbar definieren? Microsoft gibt hier mit dem Open Data Protocol (OData) eine ausführliche und standardisierte Antwort.

Die Geschäftslogik

Alle Beispiele in diesem Artikel basieren auf einer simplen Geschäftslogik mit zwei Entitäten. Die technische Grundlage bildet das Entity Framework in der Version 6. Zugrunde liegt der Code-First-Ansatz. Die vom Entity Framework erzeugten Instanzen sollen auch gleichzeitig die Geschäftsobjekte repräsentieren. Zu beachten ist, dass die feste Verdrahtung der Geschäftslogik mit einem objektrelationalen Mapper bei einer größeren Anwendung sorgfältig geprüft werden sollte. Für eine Beispielanwendung ist dies aber kein Problem. Es gibt somit die Entität *Customer*, die eine beliebige Anzahl an Rechnungen besitzen kann, zu sehen in Listing 1.

Daten per Web API abrufen

Als erster Anwendungsfall soll eine Liste von Kunden angezeigt werden. Für diese häufig benötigte Aufgabe existiert sogar eine Scaffolding-T4-Vorlage in Visual Studio 2013 unter der Bezeichnung *Web API 2 Controller with actions, using Entity Framework*, siehe Bild 1.

Visual Studio generiert dabei einen Controller, der über das ASP.NET Web API das Erzeugen, Lesen, Ändern und Löschen (CRUD) der Entität *Customer* ermöglicht. Diese atomaren Operationen entsprechen den HTTP-Verben *POST*, *GET*, *PUT* und *DELETE*. Folgender Aufruf zum Beispiel gibt eine Liste von Kunden zurück:

```
GET http://example.org/api/Customers
```

Passend dazu zeigt der Ausschnitt aus Listing 2 die von Visual Studio generierte *Code-first*Methode.

Dieser Web-API-Controller lässt sich über den *$http*-Dienst von AngularJS aufrufen. Der Dienst akzeptiert einen String oder ein Konfigurationsobjekt. Der Rückgabewert der Me-

Listing 1: Die Geschäftslogik in der Klasse Customer

```
public class Customer
{
  public Customer()
  {
        Invoices = new List<Invoice>();
  }

  public int Id { get; set; }
  public string FirstName { get; set; }
  public string LastName { get; set; }
  public string Mail { get; set; }
  public DateTime DateOfBirth { get; set; }
  public virtual ICollection<Invoice>
    Invoices { get; set; }
}

public class Invoice
{
  public int Id { get; set; }
  public decimal Amount { get; set; }

  public int CustomerId { get; set; }
  public virtual Customer Customer { get; set; }
}

public class DataContext : DbContext,
    IDataContext
{
  public DbSet<Customer> Customers { get; set; }
  public DbSet<Invoice> Invoices { get; set; }

  protected override void OnModelCreating(
    DbModelBuilder modelBuilder)
  {
    modelBuilder.Configurations.Add(new
      InvoiceMap());
  }
}
```