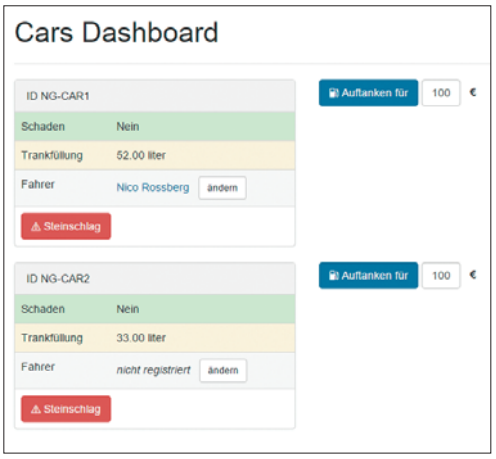
ROUTING MIT ANGULAR 2.0

Seitenwechsel ohne merkbaren Ladevorgang

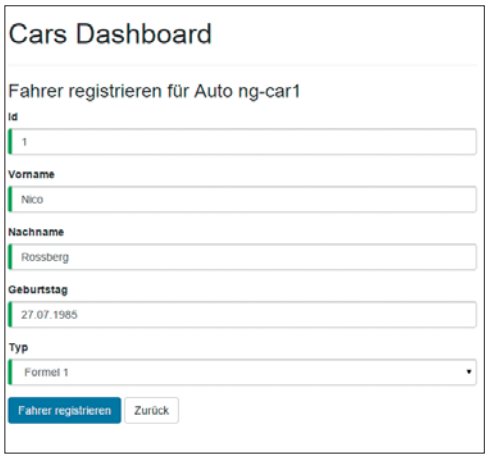
Zwischen verschiedenen Ansichten einer Single-Page-Anwendung navigieren.

Bei der Navigation in Single-Page-Anwendungen profitiert der Nutzer von Seitenwechseln ohne merkbaren Ladevorgang. Das Routing wurde in Angular 2 intensiv ausgebaut und deckt nun auch fortgeschrittene Anwendungsfälle ab.

Dies ist der fünfte und letzte Artikel aus unserer Reihe zu Angular 2. In den vorherigen Artikeln haben wir bereits SystemJS, Templates, Dependency Injection, Unit-Testing, HTTP-Kommunikation und die Verarbeitung von Formulardaten kennengelernt. Mit dabei ist stets das Car Dashboard, das kontinuierlich um neue Funktionen erweitert wird. In diesem Artikel wollen wir alle Bereiche der Anwendung über die neue Routing-Engine miteinander kombinieren (Bild 1, Bild 2).



Erste Komponente: Das Cars Dashboard mit allen verfügbaren Funktionen (Bild 1)



Zweite Komponente: Formular zum Eintragen der Fahrerdaten (Bild 2)

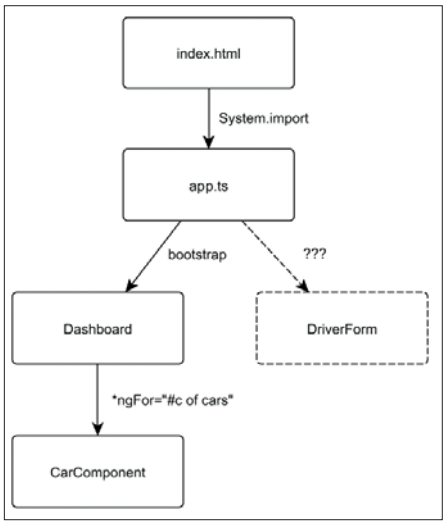
Wie gewohnt steht ein komplettes, lauffähiges Beispiel auf GitHub zur Verfügung. Sie finden alle besprochenen Inhalte unter https://github.com/Angular2Buch/angular2-routing.

Komponenten

Zugegeben, in der vorangegangenen Ausgabe mussten wir ein wenig schummeln. Wir haben neben der Komponente Dashboard eine zweite Komponente namens DriverForm vorgestellt und anhand dieser die Formularverarbeitung erläutert. Das Problem war: Beide Komponenten waren jeweils einzeln in die Website eingebunden (Listing 1).

Die hierarchische Anordnung der Komponenten aus der vorangegangenen Ausgabe sehen wir in Bild 3.

Nun wollen wir natürlich auch in der Lage sein, beide Ansichten gleichzeitig zu verwenden, damit wir durch die Anwendung navigieren können. Hier kommt das Prinzip des Routings in Spiel. Als Routing bezeichnen wir das Laden von Bereichen der Anwendung abhängig vom Zustand.



Ohne Routing kommt man hier nicht weiter (Bild 3)

Im Prinzip geschieht dasselbe, was wir auch manuell gemacht haben: Komponenten werden ausgetauscht. Der Dienst, der den Zustand der Angular-Anwendung verwaltet, nennt sich Router. Mittels Routing wollen wir nun Dashboard, Registrierungs-Formular und eine Detailansicht erreichbar machen. Alle Ansichten sollen vom Nutzer über verschiedene URLs aufrufbar sein.

Das Prinzip der Single-Page-Applikation sieht eine einzige HTML-Seite vor, deren tatsächliche Inhalte asynchron nachgeladen werden. Dabei findet in der Regel kein Neuladen der Seite statt. Das HTML5 History API liefert die technische Grundlage, um das Routing adäquat anzugehen.

Für ältere Browser, die dieses API nicht unterstützen, existieren Fallbacks, wie zum Beispiel die Verwendung von URLs mit einem *#Hash* (HashLocationStrategy). Der in Angular 2 standardmäßig vorhandene Rou-