Ein ungleiches Paar

Java oder JavaScript?

Stephan Rauh

Sind Sie Java-Programmierer? Dann fühlen Sie sich bestimmt auch manchmal, als lebten Sie in einem abgelegenen Tal. Manchmal hört man Geräusche: Das ist der Technologiezug, den man in der Ferne sieht, wenn man auf den Berg läuft. Wo mag er hinfahren? Kollegen mit scharfen Augen behaupten, sie könnten das Wort „JavaScript" auf dem Schild mit dem Reiseziel lesen. Ihr Fernweh ist unübersehbar. Und, ach, es steckt an ...

Als ich neulich solch einen Kommentar auf meiner Webseite fand, musste ich schmunzeln. So geht es mir doch auch! Und doch gefällt es mir in meinem „Tal". Und ich denke gar nicht daran wegzuziehen. So schlecht finde ich Stabilität nicht. Ist Verlässlichkeit nicht eine Tugend, solange sie nicht in Stagnation umschlägt? Was fühlten wir uns abgehängt mit unseren Applets, damals, als der Struts-Zug vorbeifuhr! Und dann kam Spring, was sich damals wie die Erfindung der Einfachheit anfühlte. Gar kein Vergleich zu EJB 2.0. Kurz darauf kam Ruby on Rails, und schon fühlte sich Spring alt an. Bis Grails kam. Und GWT. Also Java, das Dank Googles Magie im Browser ausgeführt wird. Dann kamen ExtJS, Meteor, AngularJS, …

Der kometenhafte Aufstieg von JavaScript

Es ist schon unglaublich, welchen Aufstieg JavaScript erlebt hat. Eine Suche auf Google Trends zeigt es deutlich: *"undefined is not a function"* wurde 2014 aus dem Stand heraus eine sehr populäre Frage. Diese Fehlermeldung ist das JavaScript-Gegenstück zu Javas `NullPointerException` und zeigt daher sehr schön, wie populär JavaScript in den letzten Jahren geworden ist. Mittlerweile gehört es zum guten Ton, als Java-Entwickler jQuery zu beherrschen. Inzwischen erlauben die beiden Projekte Node.js und Nashorn sogar den Einsatz von JavaScript auf dem Server.

Wozu JavaScript?

Vor zwei Jahren stieß ich auf einige Herausforderungen, die sich mit einem traditionellen JSF-Projekt nur schwer lösen ließen. In seinen Anfangstagen war JSF ein einfaches Request-Response-Framework. Die Anwendung reagierte nur auf Klicks, die auf Buttons oder Menüpunkten ausgeführt wurden. Eine klassische Windows-Anwendung hingegen bietet ein reichhaltiges Angebot an Events, auf die der Programmierer reagieren kann, zum Beispiel wenn ein Feld betreten oder verlassen wird, oder wenn eine Checkbox angeklickt wird oder ein bestimmtes Feld verändert wird. Besonders knifflig wird es, wenn mehrere Felder voneinander abhängen, jedes Feld editierbar ist und sich die Änderung in jedem dieser Felder sofort auf die Werte der anderen Felder auswirken soll. Denken Sie an zwei Prozentsätze, die zusammen 100 Prozent ergeben sollen. Wenn eines der Felder verändert wird, soll das andere Feld automatisch neu berechnet werden.

Ein klassisches Request-Response-Framework ist mit dieser Aufgabe überfordert, weil es nur schwer herausfinden kann, welches der beiden Felder gerade verändert wurde und welcher Prozentsatz neu berechnet werden soll. Ajax kann dieses Problem lösen. Leider erfordert die Lösung in JSF viel Programmcode und ist immer noch nicht perfekt. Bei langsamen Mobilfunknetzen macht sich diese Latenz unangenehm bemerkbar: Wenn man nicht sehr aufpasst, geht der Eingabefokus verloren und die nächste Eingabe des Anwenders geht ins Leere.

Da ist es schon besser, die Aufgabe mit wenigen Zeilen JavaScript zu lösen. JavaScript kann auf alle denkbaren Benutzeraktionen reagieren – sogar auf Wisch-Gesten.

Komplett auf JavaScript umsteigen?

Konsequenterweise könnte man also Anwendungen komplett in JavaScript schreiben. Heutzutage sind die Browser so mächtig geworden, dass das in den meisten Fällen problemlos geht und häufig auch gemacht wird.

Java spielt in diesem Szenario also nicht mehr die Hauptrolle. JavaScript-Anwendungen kommunizieren beispielsweise nur noch mit einer REST-Façade, hinter der sich ein Java-Service verbergen kann – oder auch direkt ein SAP-Service.

Der Erfolg von JavaScript hat jedoch auch Schattenseiten: Fast wöchentlich – so scheint es – kommen neue Frameworks heraus. Und das Tempo beschleunigt sich noch. Wie schrieb neulich jemand auf Twitter? „Wenn es so weitergeht, ist das nächste hochgelobte JavaScript-Framework bald drei Tage nach dem Release veraltet" [Neck].

Zeit, innezuhalten

Lohnt es sich wirklich, jedem Trend nachzujagen? Wozu soll das gut sein? Der Umstieg auf ein neues Framework wird oft damit gleichgesetzt, die Anwendung neu zu schreiben. In den meisten Fällen ist das aber weder sinnvoll noch notwendig. Es gibt viele Möglichkeiten, das „coole Neue" in die bestehende, bewährte Anwendungslandschaft zu integrieren.

Im Folgenden stelle ich einige Möglichkeiten vor, Java mit JavaScript zu kombinieren. Dabei entwickle ich ein kleines Beispiel, bei dem mehrere unterschiedliche Frameworks zum Einsatz kommen, damit Sie die verschiedenen Ansätze und die unterschiedlichen Programmierstile vergleichen können. Sie finden die Quelltexte auf dem Server von JavaSPEKTRUM [SIGS] sowie auf GitHub [Quelltext1].

Die Wahl der Frameworks ist subjektiv gefärbt: Ich lege meinen Schwerpunkt auf die Frameworks, mit denen ich mich am besten auskenne oder die ich aus anderen Gründen als für Enterprise-Entwickler interessant ansehe.

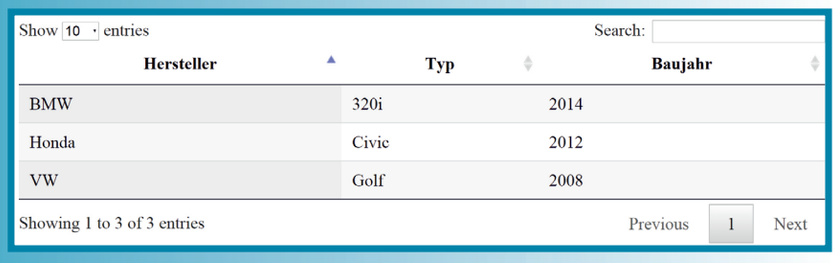

Abb. 1: Tabelle mit jQuery

Actionorientierte Frameworks wie Spring MVC und Ozark

Viele Webframeworks bieten nur eine relative dünne Abstraktionsschicht zwischen dem Browser und dem Programm. Das gilt vor allem für actionorientierte Frameworks wie Spring MVC und Ozark. Bei Servlets und JSP-Seiten haben Programmierer eine weitgehende Kontrolle über alle Aspekte der HTML-Seite. Bei Spring MVC sieht es ähnlich aus. Falls Sie keine umfangreiche Komponentenbibliothek verwenden, stellt sich die Frage „Java oder JavaScript" kaum: Die Seite können Sie sehr schnell um eine JavaScript-Funktionalität bereichern. Dafür gibt es zahlreiche jQuery-Plug-ins, mit denen Sie Ihre Anwendung aufpeppen können.

Wie einfach das geht, zeige ich gleich an einer HTML-Seite (s. Abb. 1, [Quelltext1]). Die Idee lässt sich leicht auf HTML-orientierte Frameworks wie JSP, Spring MVC und Ozark übertragen.

Ein schönes Beispiel für den lohnenden Einsatz von JavaScript sind Tabellen mit Sortierfunktion und Paginator. Das traditionelle Request-Response-Schema erzeugt HTML-Quelltexte der Tabelle vollständig auf dem Server. Das sind oft sehr große Seiten, die zum Client geschickt werden. Beim Sortieren oder Blättern wird die Aktion auf dem Server ausgeführt.

Anschließend wird abermals der HTML-Code generiert und zum Client geschickt. Die neu generierte Seite unterscheidet sich oft nur in wenigen Details von der vorher generierten Seite. Ein großer Teil des HTML-Codes wird doppelt generiert und über das Netzwerk verschickt, was zulasten der Performance geht. In vielen Fällen ist es geschickter, lediglich die Daten, die die Tabelle enthält, zum Client zu schicken, und das Rendern, Sortieren und Paginieren dem Client zu überlassen. [Quelltext1] verwendet dafür ein jQuery-Plug-in. Abbildung 1 zeigt, dass das Beispiel schon mit den Default-Einstellungen ansprechend aussieht.

Um das Beispiel einfach zu halten, füllt das Programm ein JavaScript-Array mit festen Werten. In einer Spring-MVC-Anwendung würden Sie diese Werte aus einer Spring-Bean auslesen.

AngularJS

Wenn Sie Ihre Anwendung komplett in JavaScript entwickeln wollen, bietet sich AngularJS an [AngularJS]. Das gilt insbesondere für große Anwendungen. AngularJS ist ein clientseitiges MVC-Framework mit Dependency-Injection. Entwickler, die aus dem Enterprise-Umfeld kommen, finden in AngularJS viele gewohnte Konzepte wieder. Wie groß die Ähnlichkeit zwischen JSF und AngularJS ist, hat Rudy De Busscher auf seinem Blog beschrieben [DeBusscher13].

AngularJS-Applikationen laufen komplett auf dem Client. Dementsprechend spielt die HTML-Seite eine zentrale Rolle für das Maskendesign. Die Elemente der HTML-Seite können durch zusätzliche Elemente oder Attribute ergänzt werden, deren Bedeutung erst durch AngularJS definiert wird. In [Quelltext2] ist das beispielsweise die Direktive `ng-repeat`, die dafür sorgt, dass für jeden Eintrag in der Fahrzeugtabelle eine Tabellenzeile (also ein `<tr>`-Element) erzeugt wird.

Der Trick, Funktionalität durch zusätzliche Attribute zur Verfügung zu stellen, ist für Web-Designer hilfreich: Die Grundstruktur der HTML-Seite ist auch ohne AngularJS gut zu erkennen. Ein Web-Designer kann für seine Arbeit seine gewohnten Werkzeuge verwenden. Das ist ein großer Vorteil im Vergleich zu vielen serverbasierten Webframeworks, deren Maskendefinitionen nicht als HTML-Quelltext vorliegen, sondern bei denen der HTML-Quelltext erst durch das Framework generiert wird.

Die Verbindung der Eingabefelder zum Datenmodell von AngularJS wird durch das Attribut „ng-model" in den HTML-Tags hergestellt. Das clientseitige Datenmodell wird bereits während der Eingabe aktualisiert. Plausibilitäten werden bereits beim Verlassen der Felder auf dem Client ausgewertet, und der Klick auf die Schaltfläche „Hinzufügen" ruft die JavaScript-Methode `neuesFahrzeug()` auf, die die Variable `dataSet` im AngularJS-Model verändert. AngularJS erkennt diese Änderung und aktualisiert die angezeigte Tabelle, ohne dass Sie als Programmierer aktiv werden müssten.

Die Funktionalität der Anwendung wird mit Hilfe eines AngularJS-Controllers mit der HTML-Seite verknüpft. In [Quelltext3] bewirkt das Attribut `ng-click` im Button-Tag, dass beim Klick auf die Schaltfläche die Methode `neuesFahrzeug()` des Controllers aufgerufen wird, wodurch wiederum eine neue Zeile in der Tabelle hinzugefügt wird.

Besonders interessant wird AngularJS durch die Möglichkeit, Komponenten zu bilden. In der Praxis fühlt sich das so an, als könnte man neue HTML-Tags definieren. Die Tabelle, für die wir in [Quelltext2] die Zeilen 8 bis 21 gebraucht haben, benötigt bei der Verwendung der Bibliothek AngularPrime zum Beispiel nur noch eine einzige Zeile HTML-Quelltext:

```
<div pui-datatable="autos.dataSet"></div>
```

Das ist leichter lesbar und damit besser zu warten. Mittlerweile gibt es viele weitere interessante Komponentenbibliotheken für AngularJS, wie ng-table, smart-table und seit einiger Zeit auch Kendo UI. Die Links zu den Projekten finden Sie weiter unten im Literaturverzeichnis ([ngTable], [smartTable] und [KendoUI]).

Sie sollten aber die Vor- und Nachteile von fertigen Komponentenbibliotheken genau abwägen: Passt die Komponente nicht genau, sondern nur fast auf Ihre Anforderungen, beginnen Sie unter Umständen schnell, gegen die Limitationen der Komponente zu kämpfen. Das kostet Sie am Ende unnötig Zeit, anstatt bei der Produktion zu helfen.

JavaServer Faces

JSF war ursprünglich ein klassisches Request-Response-Framework. Es stammt noch aus der Zeit, als die direkte Programmierung im Browser mit Hilfe von JavaScript und HTML schwierig war. JSF generiert daher den HTML-Quelltext serverseitig. Besonders komfortabel wird JSF durch die Verwendung von Komponentenbibliotheken wie PrimeFaces, RichFaces, ICEfaces oder BootsFaces. Die Verwendung der Komponenten gestaltet sich ähnlich wie bei AngularJS. Der Unterschied ist, dass der HTML-Quelltext nicht auf dem Client, sondern auf dem

Server generiert und über das Netzwerk verschickt wird. Die generierten HTML-Quelltexte können sehr umfangreich sein und eine hohe Last auf Server und Netzwerk erzeugen.

Viele JSF-Komponentenbibliotheken sind seit Jahren auf dem Markt und entsprechend ausgereift. Sie bieten oft überraschende Features. PrimeFaces legt zum Beispiel großen Wert auf die Unterstützung von Screen-Readern für Sehbehinderte und erlaubt dem Anwender, in vielen Fällen ohne Verwendung der Maus nur mittels Tastatur zu arbeiten.

Andererseits gelten die schon bei AngularJS angesprochenen Probleme auch bei JSF-Komponenten: Wenn Sie einen anspruchsvollen Auftraggeber haben, erfüllen vorgefertigte Komponenten die Anforderungen eventuell nicht ganz. Sie laufen dann Gefahr, gegen die Einschränkungen der Komponente „anzuprogrammieren".

JavaScript in den JSF-Implementierungen

Interessant wird es, wenn Sie sich den Aufbau einer PrimeFaces-Komponente genauer anschauen. Sie besteht nämlich zu einem großen Anteil aus JavaScript. Im Prinzip ermöglicht JSF es Ihnen, Webanwendungen zu programmieren, ohne tiefere Kenntnisse von HTML, JavaScript und CSS zu haben. Dennoch bestehen Ihre Programme am Ende zum Teil aus JavaScript.

Manchmal macht sich das unangenehm bemerkbar. Wenn Ihnen bei der Definition der JSF-Seite ein Fehler unterläuft, begrüßt Sie gelegentlich die Fehlermeldung *"undefined is not a function"*. Das Problem an der Sache ist, dass Sie als JSF-Spezialist mit dieser Fehlermeldung wenig anfangen können. Um den Fehler zu finden, müssen Sie sich sowohl mit dem JavaScript-Quelltext als auch mit dem Java-Quelltext der JSF-Komponente beschäftigen.

Dass die meisten Teams nach der Suche auf StackOverflow.com die Methode „Versuch und Irrtum" vorziehen dürften, spricht eigentlich dafür, die Anwendung gleich in JavaScript zu entwickeln, insbesondere da Sie als JSF-Entwickler auf Dauer ohnehin nicht daran vorbeikommen, sich intensiv mit HTML, CSS und JavaScript zu beschäftigen.

Asynchronous JavaScript and XML mit JSF

Ajax wurde nachträglich zu JSF hinzugefügt. Bei allen Vorteilen, die das hatte, brachte dies aber auch mehr Arbeit für den JSF-Entwickler mit sich. Er musste zum Beispiel ab jetzt für jeden einzelnen Ajax-Request festlegen, welcher Teil der Seite aktualisiert werden soll. ICEfaces [ICEfaces] und BabbageFaces [BabbageFaces] zeigen, wie es besser geht: Die JSF-Frameworks können selbstständig ermitteln, welche Teile der Seite verändert wurden, und aktualisiert dementsprechend nur einen Teil der Seite.

Noch einfacher ist es, gleich auf Ajax zu verzichten und die Logik stattdessen auf den Client zu verlagern. Im Abschnitt „Wozu JavaScript?" wurde schon beschrieben, dass dadurch manche Anforderungen leichter implementiert werden können. Die meisten Funktionalitäten einer Webseite, die durch einen Ajax-Request implementiert werden, benötigen eigentlich keine Zugriffe auf Backend-Systeme, sondern könnten mit JavaScript auf dem Client implementiert werden. Natürlich gibt es auch Fälle, in denen eine Funktion den Backendzugriff braucht, das sind aber seltene Ausnahmen.

JSF und insbesondere PrimeFaces bieten ein umfangreiches JavaScript-API. Wenn Sie sich ein wenig mit jQuery und dem DOM-Baum der HTML-Seite auskennen, spricht nichts gegen den Einsatz von JavaScript in Ihrer JSF-Anwendung.

AngularFaces: JSF plus AngularJS

Um die Integration von JSF und JavaScript noch weiter zu vereinfachen und um die Generierung von Komponenten vom Server auf den Client zu verlagern, habe ich AngularFaces entwickelt [AngularFaces]. Das „Model" von AngularJS wird automatisch mit den Beans von JSF synchronisiert, sodass Sie in vielen Fällen die freie Wahl haben, ob Sie eine Funktionalität in JavaScript oder Java entwickeln. Das gibt Ihnen die Freiheit, das Programm durch den Verzicht auf Ajax zu vereinfachen. Oder Sie verwenden Ajax weiterhin, aber seltener, und verbessern dadurch die User-Experience. Der Programmablauf fühlt sich dann schneller an, und Server und Netzwerk werden entlastet.

Sie können AngularFaces auch als Brückentechnik einsetzen. Große, über die Jahre gewachsene JSF-Anwendungen können oft nicht von heute auf morgen durch AngularJS-Anwendungen ersetzt werden. AngularFaces erlaubt es Ihnen, mit sehr geringem Risiko Erfahrungen mit AngularJS zu machen. [Quelltext4] kombiniert AngularFaces mit dem AngularJS-Widget Smart-Table, das in etwa vergleichbar ist mit der DataTable von PrimeFaces. Hinzu kommt noch die JSF-Bibliothek BootsFaces, um die Seite mit dem Look-and-Feel und dem Design von Bootstrap anzuzeigen (s. Abb. 2). BootsFaces vereinfacht die Entwicklung einer Bootstrap-Seite [BootsFaces]. Beispiele sind die Tags `container`, `row` und `column`, die kürzer und lesbarer sind als die entsprechenden `div`-Tags von Bootstrap.

Der AngularJS-Controller [Quelltext5] ist weitgehend mit dem Controller der reinen AngularJS-Lösung identisch. Der Unterschied besteht hauptsächlich darin, dass das AngularJS-

Hersteller	Type	Baujahr
VW	Golf	2008
Honda	Civic	2012
BMW	320i	2014
Fiat	Punto	2005

Hersteller: Renault | Typ: Megane | Baujahr: 2011

hinzufügen

Abb. 2: Tabelle mit AngularFaces und der AngularJS-Komponente Smart-Table

Model diesmal durch den Aufruf der Methode `initJSFScope()` von der JSF-Bean befüllt wird.

HTML5-orientiertes JSF

Es geht auch einfacher – und das kommt vor allem jenen JSF-Teams entgegen, die einen Webdesigner an Bord haben. Für einen Webdesigner sind klassische, schwergewichtige JSF-Komponenten ausgesprochen lästig. Seine Werkzeuge kommen mit JSF nicht zurecht. Seit JSF 2.2 können Sie mit wenig Aufwand ganz normale Webseiten in JSF-Seiten verwandeln. Sie brauchen nur die Eingabefelder und die Buttons, um ein JSF-spezifisches Attribut zu ergänzen. In diesem Fall können Sie alle Register von CSS und JavaScript ziehen. Wenn Sie AngularJS-Komponenten einsetzen wollen, hilft Ihnen die Bibliothek JUA von Marco Rinck [Rinck14], die die Zusammenarbeit von JSF und AngularJS bei Ajax-Anwendungen verbessert beziehungsweise teilweise überhaupt erst ermöglicht.

PrimeUI und AngularPrime

PrimeFaces bietet einige interessante Ableger für JavaScript-Entwickler. PrimeUI ist eine reine JavaScript-Variante von PrimeFaces. Sie können also eine Anwendung mit dem Look-and-Feel von PrimeFaces schreiben, die komplett auf dem Client läuft (siehe [PrimeUI]).

AngularPrime kombiniert AngularJS mit PrimeUI. Sie können damit AngularJS-Anwendungen schreiben, die sich nahtlos in bereits bestehende serverseitige PrimeFaces-Anwendungen integrieren (siehe [AngularPrime]).

Scala.js, Dart und andere Sprachen auf dem Client

Auch auf dem Client gibt es Alternativen zu JavaScript. Dart ist eine Programmiersprache, die nach JavaScript kompiliert wird, aber einen ähnlichen Programmierstil bietet wie ein (modernisiertes) Java. Für Dart gibt es einige Frameworks, die die Programmierung weiter vereinfachen, beispielsweise AngularDart und Polymer.dart (siehe [AngularDart] und [PolymerDart]).

Scala.js entspricht Scala mit der Besonderheit, dass es nach JavaScript statt in Java-Byte-Code kompiliert. Wenn Sie Scala.js ausprobieren wollen, schauen Sie sich das Scala.js-Fiddle an [ScalaJS]. Es bietet Ihnen eine einfache, aber vollständige Entwicklungsumgebung im Browser, komplett mit einigen Beispielprogrammen.

Ähnliche Ideen wie Scala.js gibt es auch bei anderen Programmiersprachen. Beispielsweise gibt es ClojureScript für Clojure-Entwickler und PureScript für Freunde von Haskell.

Fazit und Ausblick

Es gibt also zahlreiche Varianten, Java und JavaScript zu kombinieren. Mittelfristig scheint der Trend klar zu reinen JavaScript-Anwendungen zu gehen. Hier ist für jeden Geschmack etwas dabei: Neben den hier beschriebenen, größtenteils schwergewichtigen Enterprise-Frameworks gibt es auch zahlreiche leichtgewichtige Frameworks. Man kann aber bestehende Java-Anwendungen auch durch JavaScript auffrischen.

Ach ja, und was das „abgelegene Tal" betrifft: Wie es scheint, sitzen die Java-Programmierer längst im Zug und fahren in das Land der JavaScripte!

JavaSPEKTRUM ist eine Fachpublikation des Verlags:
SIGS DATACOM GmbH
Lindlaustraße 2c • 53842 Troisdorf
Tel.: 0 22 41 / 23 41-100 • Fax: 0 22 41 / 23 41-199
E-Mail: info@sigs-datacom.de • www.javaspektrum.de
SIGS DATACOM
FACHINFORMATIONEN FÜR IT-PROFESSIONALS

Links

[AngularDart] https://angulardart.org/
[AngularFaces] https://www.angularfaces.net
[AngularJS] https://angularjs.org/
[AngularPrime] http://angularprime.appspot.com/#/main
[BabbageFaces] http://www.beyondjava.net/blog/introducing-babbage-faces-efficient-ajax-dirt-cheap/
[BootsFaces] http://www.bootsfaces.net/
[DeBusscher13] R. De Busscher, Comparison between AngularJS and JSF, 21.1.2013, http://statelessprime.blogspot.de/2013/01/comparison-between-angularjs-and-jsf.html
[ICEfaces] http://www.icesoft.org/wiki/display/ice/automatic+ajax
[KendoUI] http://www.telerik.com/kendo-ui
[Neck] https://twitter.com/neckbeardhacker/status/569883173531873280
[ngTable] http://ng-table.com/
[PolymerDart] https://www.dartlang.org/polymer/
[PrimeUI] http://www.primefaces.org/primeui/
[Quelltext1] https://github.com/stephanrauh/BeyondJava.net-Articles/blob/master/CombiningJavaAndJavascript/src/main/webapp/clientSideTable.html
[Quelltext2] https://github.com/stephanrauh/BeyondJava.net-Articles/blob/master/CombiningJavaAndJavascript/src/main/webapp/AngularJSTable.html
[Quelltext3] https://github.com/stephanrauh/BeyondJava.net-Articles/blob/master/CombiningJavaAndJavascript/src/main/webapp/AngularJSTable.js
[Quelltext4] https://github.com/stephanrauh/BeyondJava.net-Articles/blob/master/CombiningJavaAndJavascript/src/main/webapp/angularFaces.xhtml
[Quelltext5] https://github.com/stephanrauh/BeyondJava.net-Articles/blob/master/CombiningJavaAndJavascript/src/main/webapp/cars.js
[Rinck14] M. Rinck, How to use angularJS directives to replace JSF components, 3.11.2014, http://entwicklertagebuch.com/blog/2014/11/jsf-updates-angular-how-to-use-angularjs-directives-to-replace-jsf-components/
[ScalaJS] http://www.scala-js-fiddle.com/gist/
[SIGS] Quellen zum Artikel, http://www.sigs-datacom.de/nc/fachzeitschriften/javaspektrum/archiv.html
[smartTable] http://lorenzofox3.github.io/smart-table-website/

Stephan Rauh arbeitet als Senior Consultant bei der OPITZ CONSULTING Deutschland GmbH am Standort Bad Homburg. Seit mehreren Jahren befasst er sich mit JSF und AngularJS und ist in der Open-Source-Szene als Autor von AngularFaces und Committer für BootsFaces engagiert.
E-Mail: Stephan.Rauh@opitz-consulting.com