Frontend-entwicklung mit AngularJS

Rene Behring & Damian Poddebniak

7. Januar 2015

Zusammenfassung

Ziel dieses Praktikums ist die Entwicklung eines Frontends mit AngularJS. Die Web-Anwendung ermöglicht im späteren Verlauf die Interaktion mit einem Web-Service über einen herkömmlichen Browser.

Der zu verwendende Web-Service ist eine leicht modifizierte Variante des im letzten Praktikum entstandenen JAX-RS Projektes.

Im Laufe des Praktikums lernen Sie neben AngularJS auch eine neue Entwicklungsumgebung (WebStorm) und eine bisher im Studium nicht behandelte Programmiersprache (JavaScript) kennen.

Am Ende des Aufgabenzettels finden Sie ein paar nützliche Literatur- und Dokumentationshinweise.

1 Einrichtung

Die empfohlene Entwicklungsumgebung für AngularJS Projekte ist *WebStorm*. Die kostenlose Version ist auf der *JetBrains* Homepage unter `http://www.jetbrains.com/webstorm/` verfügbar und kann 30 Tage lang getestet werden.

1. Laden Sie die neuste WebStorm Version herunter und starten Sie die Entwicklungsumgebung. Unter Linux kann dazu einfach das Skript `bin/webstorm.sh` ausgeführt werden.

2. Checken Sie das zu bearbeitende Projekt mit dem Kommando

   ```
   git clone https://github.com/Duesentrieb-Incorporation/angular-praktikum
   ```

   über die Konsole aus. In dem Ordner befinden sich das Front- sowie auch das Backend.

3. Importieren Sie das AngularJS Projekt in WebStorm. Klicken Sie dazu einfach auf *Open* und navigieren Sie zu dem Projektordner mit dem Namen `frontend`. Schauen Sie sich die Seite im Browser an. Klicken Sie dazu über `frontend/app/index.html` auf *Run 'index.html'*.

   Hinweis: Über *File* ⇒ *Settings...* ⇒ *Tools* ⇒ *Web Browsers* können Sie einen anderen Browser zum starten konfigurieren.

4. Importieren Sie das JAX-RS Maven-Projekt in Eclipse. Klicken Sie dazu einfach auf *File* ⇒ *Import* ⇒ *Existing Maven Project* und navigieren Sie zu dem Projektordner mit dem Namen `backend`. Starten Sie den Server wie gewohnt.

Die Aufgaben beziehen sich ausschließlich auf das Frontend. Das Backend stellt uns lediglich Daten für unsere Web-Anwendung bereit. Es ist nicht notwendig etwas im Backend zu ändern.

2 Willkommensseite

1. Verschaffen Sie sich zunächst einen Überblick über das Projekt. Welche Module werden verwendet? Wie sind die Controller, Views und Adressen verknüpft? Es genügt wenn Sie sich zunächst die Inhalte der beiden Ordner `home` und `articles` anschauen.

2. Korrigieren Sie die Willkommensnachricht. Wie wird mit dem Modell gearbeitet und wie greifen Sie auf den "scope" zu?

3. Injecten Sie im `HomeController` Angulars `$log` Service und geben Sie eine beliebige Meldung auf der Konsole aus. Wann wird der `HomeController` initialisiert?

Tipp: Sie können den `$log` Service in jedem Controller injecten. Dies ist insbesondere hilfreich um sicherzustellen das auch die richtigen Controller benutzt werden.

3 Öffentliche Artikelansicht

Ziel dieser Aufgabe ist es die Nutzung verschiedener Direktiven und die Konfiguration des Routings zu vermitteln. Die öffentliche Artikelansicht besteht aus eine verkürzte Liste vorhandener Artikel (Masterview) und einer detaillierten Ansicht eines einzelnen Artikels (Detailview).

1. Legen Sie im `ArticleController` (und nur hier) eine Liste von Artikeln an und erweitern Sie das Template in `articles/articles.html` um Direktiven, sodass Ihre Liste in der Website gezeigt wird. Ein Artikel hat die Attribute `title` und `content`.

2. Limitieren Sie den gezeigten Artikelinhalt auf 100 Zeichen und vervollständigen Sie die Filterfunktion. Fügen Sie einen Link zum weiterlesen eines ausgewählten Artikels ein. Nutzen Sie als Artikel-ID zunächst einfach den Iterationsindex $index.

3. Erweitern Sie das Routing so, dass beim Aufruf eines speziellen Artikels der `ArticleDetailController` als Controller und `articles-detail.html` als Template verwendet werden.

4. Injecten Sie im `ArticleDetailController` Angulars `$routeParams` Service und geben Sie die Artikel-ID über den `$log` Service auf der Konsole aus. Der Artikel muss hier noch nicht angezeigt werden können.

5. Vervollständigen Sie den `ArticleMock` Service. Dieser ist definiert in `services/article-service-mock.js`. Refactoren Sie anschließend Ihren Code in beiden Artikel-Controllern um nun über den Service an die Artikel zu gelangen.

6. Rufen Sie im `ArticleDetailController` den passenden Artikel zur gewählten URL ab und zeigen Sie ihn in der Detailview an. Prüfen Sie bevor Sie mit der nächsten Aufgabe fortfahren, ob die Master- und Detailview korrekt zusammen funktionieren.

4 Kommunikation mit dem Server

In dieser Aufgabe wird die Kommunikation mit dem Backend über Angulars `$resource` Service hergestellt.

1. Ändern Sie den `ArticleController` so, dass der `Article` Service benutzt wird, anstatt des `ArticleMock` Services.

2. Die Master- und Detailaview verwendeten vorher den Index eines Artikels. Diese Lösung ist nun nicht mehr ausreichend. Nutzen Sie statt der Indices nun die richtige ID der Artikel. Diese schickt der Server selbstverständlich mit.

5 Administrationsoberfläche

Die Administrationsoberfläche ist der Masterview ähnlich, leitet aber zusätzlich auf entsprechende Seiten zum Hinzufügen und Bearbeiten von Artikeln weiter und erlaubt das Löschen einzelner Artikel.

1. Fragen Sie erneut alle Artikel vom Server ab und benutzen Sie den `$log` Service um Meldungen über etwaige gescheiterte Abfragen zu erhalten. Nutzen Sie ab jetzt das `$promise` Konzept für jede Anfrage wie in der Vorlesung besprochen.

2. Erweitern Sie den `AdminController` so, dass das Löschen von existierenden Artikeln ermöglicht wird. Die entsprechende Direktive (`ng-click`) ist bereits im `admin/admin.html` Template definiert.

   Tipp: Sie können `splice()` benutzen um ein Element aus einer Liste zu entfernen.

3. Erweitern Sie den `AdminEditController` so, dass die Bearbeitung von existierenden Artikeln ermöglicht wird.

   Tipp: Wenn ein Artikel erfolgreich erstellt wurde können Sie mittels

   ```
   $location.path("/admin/articles")
   ```

   den Browser zurück zur Übersicht navigieren lassen.

   (Optional)

4. Erweitern Sie den `AdminCreateController` so, dass die Erstellung von neuen Artikeln ermöglicht wird.

   Tipp: Ein einfacher POST Request genügt nicht. Das Backend erwartet eine Kombination aus POST und PUT. Das muss in der Anwendung beachtet werden.

   (Optional)

Literatur

[1] Brad Green und Shyam Seshadri. *AngularJS*. 1. Aufl. Sebastopol: O'Reilly Media, Inc., 2013. ISBN: 978-1-449-35588-3.

[2] Brad Green und Shyam Seshadri. *AngularJS: Up and Running - Enhanced Productivity with Structured Web Apps*. 1. Aufl. Sebastopol: O'Reilly Media, Inc., 2014. ISBN: 978-1-491-90192-2.

[3] Douglas Crockford. *JavaScript: The Good Parts - The Good Parts*. Sebastopol: O'Reilly Media, Inc., 2008. ISBN: 978-0-596-55487-3.

[4] *AngularJS - Superheroic JavaScript MVW Framework*. URL: https://angularjs.org/ (besucht am 12. 12. 2014).

[5] *AngularJS - ngRepeat*. URL: https://docs.angularjs.org/api/ng/directive/ngRepeat (besucht am 02. 01. 2015).

[6] *AngularJS - limitTo*. URL: https://docs.angularjs.org/api/ng/filter/limitTo (besucht am 02. 01. 2015).

[7] *AngularJS - routeParams*. URL: https://docs.angularjs.org/api/ngRoute/service/$routeParams (besucht am 02. 01. 2015).

[8] *AngularJS - resource*. URL: https://docs.angularjs.org/api/ngResource/service/$resource (besucht am 02. 01. 2015).

[9] *Promises in AngularJS: The definitive Guide*. URL: http://www.dwmkerr.com/promises-in-angularjs-the-definitive-guide/ (besucht am 02. 01. 2015).