haake & partner
– datentechnik gmbh –

DHBW
Duale Hochschule
Baden-Württemberg

Clientseitige Webframeworks wie AngularJS, ReactJS und OpenUI5

Seminararbeit

für die Prüfung zum

Bachelor of Science (B.Sc.)

des Studiengangs Wirtschaftsinformatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

Verfasser	Sebastian Greulich, Fabio Krämer
Partnerunternehmen	
Matrikelnummer, Kurs	, WWI16B2
Wissenschaftlicher Betreuer	
Abgabe	17.09.2018

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich meine Seminararbeit mit dem Thema: „*Clientseitige Webframeworks wie AngularJS, ReactJS und OpenUI5*“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, 17.09.2018

Sebastian Greulich, Fabio Krämer

Inhaltsverzeichnis

Abkürzungsverzeichnis III

Abbildungsverzeichnis IV

Listings V

1. Einleitung 1

2. Webframeworks 2

3. Beispiel Webframeworks 3

3.1. AngularJS . . . 3

3.1.1. Allgemein . . . 3

3.1.2. Konzepte . . . 3

3.1.3. Verwendung . . . 7

3.2. ReactJS . . . 7

3.2.1. Allgemein . . . 7

3.2.2. Konzepte . . . 7

3.2.3. Verwendung . . . 7

3.3. OpenUI5 . . . 7

4. Vergleich der Webframeworks 8

5. Bezug zum Projekt 9

6. Fazit 10

A. Anhang 11

Literatur 12

Abkürzungsverzeichnis

Abbildungsverzeichnis

Listings

3.1. Das Root-Module in der Datei app.module.ts 4
3.2. Implementierung einer Komponente 5

1. Einleitung

2. Webframeworks

3. Beispiel Webframeworks

3.1. AngularJS

3.1.1. Allgemein

Entwicklungsumgebung

3.1.2. Konzepte

Beispiel Angular-Anwendung.

```
├ example/
|    ├ src/
|         ├ app/
|         –app.component.css
|         –app.component.html
|         –app.component.ts
|         –app.module.ts
|         ├ assets/
|         ├ environments/
|    –index.html
|    –main.ts
|    –styles.css
|    –...
|    ├ node_modules/
|    ├ e2e/
| –...
```

Module

Eine Angular Anwendung ist modular aufgebaut und kann demnach aus mehreren Modulen bestehen. Ein Modul fasst eine zusammengehörige Codeeinheit zusammen. Ein Modul kann eine gewisse Funktionalität bereitstellen, die wiederum von anderen Modulen verwendet werden kann. (vgl. Steyer et al. 2017, S. 103 ff.)

Die Module einer Angular-Anwendung können in Root-Module, Feature-Module und Shared-Module unterteilt werden. Das Root-Modul ist einmalig in einer Angular-Anwendung vorhanden und wird beim Starten der Anwendung aufgerufen. Die Anwendung wird durch dieses Modul konfiguriert. Mithilfe von Feature Modulen kann eine Anwendung nach Anwendungsfällen gruppiert werden. Shared-Module fassen die Teile zusammen, die unabhängig vom Anwendungsfall verwendet werden können. (vgl. Freeman 2018, S. 528 ff.; vgl. *Introduction to modules*; vgl. Steyer et al. 2017, S. 105 ff.)

In der Beispielanwendung stellt die Klasse *AppModule* siehe Listing 3.1 das Root-Modul dar.

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 @NgModule({
5   imports: [ BrowserModule],
6   declarations: [AppComponent],
7   bootstrap: [AppComponent]
8 })
9 export class AppModule { }
```

Listing 3.1: Das Root-Module in der Datei app.module.ts

Komponenten und Templates

Komponenten sind Klassen, die Daten und Logik für die zugehörigen Templates bereitstellen. Diese ermöglichen die Aufteilung einer Angular Anwendung in logisch getrennte Teile. (vgl. Freeman 2018, S. 401)

Eine Komponente wird durch den Dekorator *@Component* gekennzeichnet und kann über verschiedene Dekorator-Eigenschaften (auch: Metadaten) konfiguriert werden. Die Eigenschaft *selector* identifiziert das HTML-Element, dass durch diese Komponente repräsentiert wird. Zur Anzeige der bereitgestellten Daten kann entweder ein Inline-Template *template* definiert oder auf ein externes Template *templateUrl*

verwiesen werden. (vgl. *Introduction to components*; vgl. Freeman 2018, S. 405; vgl. Steyer et al. 2017, S. 47 ff.)

Für weitere Dekorator-Eigenschaften wird auf Freeman (2018, S. 405) verwiesen. Ein Beispiel für die Implementierung einer Komponente findet sich in Listing 3.2.

```
1 import {Component} from '@angular/core';
2
3 @Component({
4 selector: hello,
5 template: `<h1>Hello World</h1>`
6 })
7 export class HelloWorldComponent {
8 }
```

Listing 3.2: Implementierung einer Komponente

Zur Darstellung von Komponenten nutzt Angular Templates. Ein Template besteht aus HTML Code erweitert um Angular Ausdrücke. Das Template kann Pipes, weitere Komponenten, Data-Binding-Ausdrücke oder Direktiven enthalten. (vgl. *Introduction to components*; vgl. Steyer et al. 2017, S. 52)

Data-Bindings stellen eine Beziehung zwischen den Daten der Komponente und einem HTML-Element her. Hierdurch kann das Aussehen, der Inhalt oder das Verhalten dieses Elements dynamisch verändert werden. (vgl. Freeman 2018, S. 237 ff.; vgl. Steyer et al. 2017, S. 52 f.)

Es können drei Arten von Bindings unterschieden werden. Die Unterscheidung erfolgt mittels der Fließrichtung der Daten.

Two-Way-Binding [()]

Property-Binding []

Event-Binding ()

Two-Way-Binding	[(target)]	aaa
Property-Binding	[target]	aaaa
Event-Binding	(target)	aaaa

Direktiven

Mit Direktiven kann einem Element zusätzliches Verhalten hinzugefügt werden. (vgl. Steyer et al. 2017, S. 265) In Angular werden folgende drei Arten von Direktiven unterschieden. (*Attribute Directives*, vgl.)

- Komponenten
- Attribut-Direktiven
- strukturelle Direktiven

Angular stellt Direktiven zur Verfügung (engl. Built-In Directives). Diese können durch eigene Direktiven erweitert werden. (vgl. Freeman 2018, S. 261)

Mit strukturellen Direktiven kann der Inhalt des HTML-Dokuments angepasst werden, indem Elemente dem diesem hinzugefügt oder entnommen werden. Hierfür verwenden die strukturellen Direktiven Templates, die beliebig oft gerendert werden. (vgl. Steyer et al. 2017, S. 269 ff.; vgl. Freeman 2018, S. 365)

Beispiele für strukturellen Direktiven aus AngularJS (vgl. Freeman 2018, S. 261 ff.):

ngIf Fügt dem HTML-Dokument Inhalt hinzu, wenn die Bedingung wahr ist.

ngfor Fügt für jedes Item einer Datenquelle den gleichen Inhalt dem HTML-Dokument hinzu.

ngSwitch Fügt dem HTML-Dokument, abhängig vom Wert eines Ausdrucks, Inhalt hinzu.

Mit Attribut-Direktiven kann das Verhalten und Aussehen des zugehörigen Elementes angepasst werden, indem Attribute hinzugefügt oder entfernt werden. (vgl. ebd., S. 339)

Beispiele für Attribut-Direktiven aus Angular-JS (vgl. ebd., S. 249 ff.):

ngStyle Mit dieser Direktive können unterschiedliche Style-Eigenschaften dem Element hinzugefügt werden.

ngClass Weißt dem Element mehrere Klassen hinzu.

Komponenten sind Direktiven mit einer eigenen View. (vgl. Steyer et al. 2017, S. 265; vgl. Freeman 2018, S. 401) Aufgrund der hohen Bedeutung der Komponente in AngularJS, wird auf diese im nächsten Gliederungspunkt näher eingegangen.

Services

3.1.3. Verwendung

Einordnung in den Kontext

3.2. ReactJS

3.2.1. Allgemein

ReactJS ist ein von Entwicklern des Unternehmens Facebook Inc. entwickeltes JavaScript Framework. **??**

3.2.2. Konzepte

Components

Lifecycle

Virtual DOM

3.2.3. Verwendung

3.3. OpenUI5

4. Vergleich der Webframeworks

5. Bezug zum Projekt

6. Fazit

A. Anhang

Literatur

Freeman, Adam (2018). *Pro Angular 6*. 3rd ed. Berkeley, CA: Apress. DOI: \url{10.1007/978-1-4842-3649-9}. URL: \url{http://dx.doi.org/10.1007/978-1-4842-3649-9}.

Google, Hrsg. *Attribute Directives*. URL: \url{https://angular.io/guide/attribute-directives}.

– Hrsg. *Introduction to components*. URL: \url{https://angular.io/guide/architecture-components}.

– Hrsg. *Introduction to modules*. URL: \url{https://angular.io/guide/architecture-modules}.

Steyer, Manfred und Daniel Schwab (2017). *Angular: Das Praxisbuch zu Grundlagen und Best Practices*. 2. Aufl. Heidelberg: O'Reilly.