



Astro

space estate

Konzept Web-Shop



Frontend

AngularJS



- Großes, zukunftssträchtiges Framework soll verwendet werden
- Angular wird in vielen Firmen/Projekten verwendet
- Vorwissen aus ersten Semestern soll erweitert werden



AngularJS Vorteile

- Teil des MEAN-Stacks
- Folgt dem MVC-Ansatz
- Geeignet für Single Page Applications
- Mehrere Personen können daran arbeiten
- Popularität nimmt stetig zu,
- wird genutzt von: Google, Nike, Forbes, Upwork, General Motors, HBO, Sony





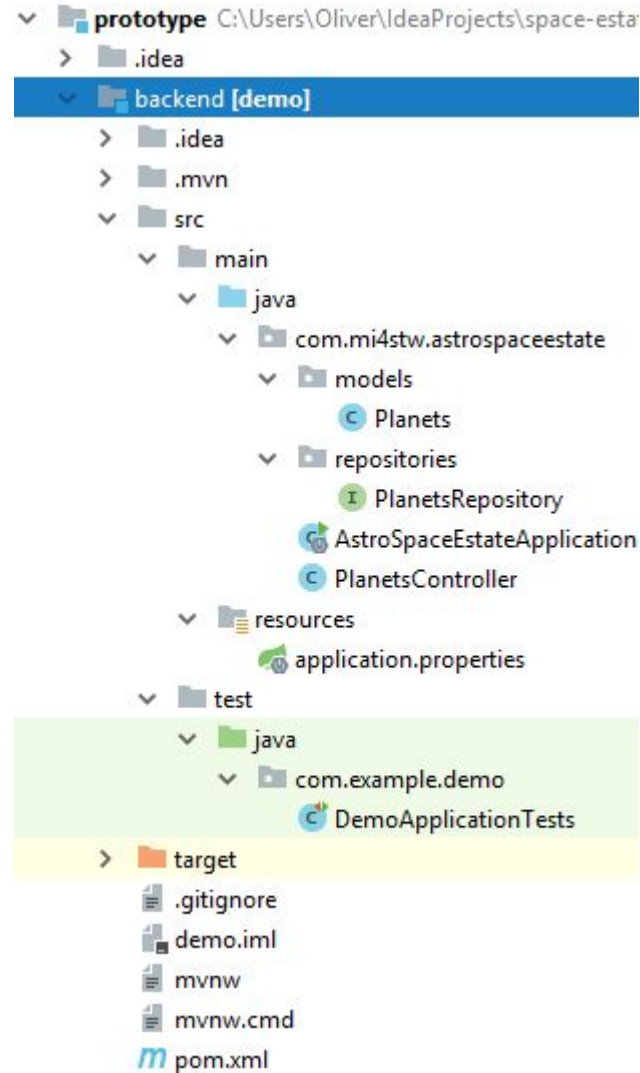
AngularJS Nachteile

- Nicht von allen Browsern unterstützt
- Alternative Frameworks für kleinere Projekte
- Einarbeiten in Angular benötigt Zeit
- Schlechte Skalierbarkeit
- Einige Probleme erst mit Angular 2.0 gelöst

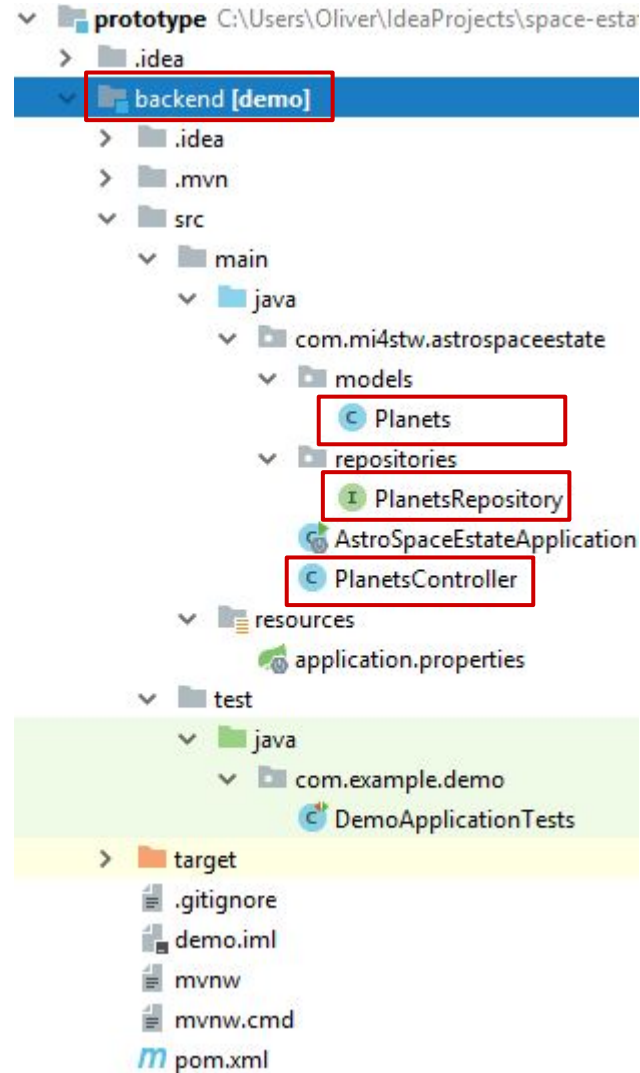


Backend

spring boot



spring boot



spring boot



PlanetsRepository

```
8  @CrossOrigin(origins = "http://localhost:4200")
9  public interface PlanetsRepository extends MongoRepository<Planets, String> {
10      Planets findBy_id(ObjectId _id);
11  }
```

spring boot



MongoRepository

```
15 @NoRepositoryBean
16 public interface MongoRepository<T, ID> extends PagingAndSortingRepository<T, ID>, QueryByExampleExecutor<T> {
17     <S extends T> List<S> saveAll(Iterable<S> var1);
18
19     List<T> findAll();
20
21     List<T> findAll(Sort var1);
22
23     <S extends T> S insert(S var1);
24
25     <S extends T> List<S> insert(Iterable<S> var1);
26
27     <S extends T> List<S> findAll(Example<S> var1);
28
29     <S extends T> List<S> findAll(Example<S> var1, Sort var2);
30 }
31
```

spring boot

Planets



spring

by Pivotal™

```
6 public class Planets {
7
8     @Id
9     public ObjectId _id;
10
11     public String name;
12     public String size;
13     public String color;
14
15     // Constructors
16     public Planets() {}
17
18     public Planets(ObjectId _id, String name, String size, String color) {
19         this._id = _id;
20         this.name = name;
21         this.size = size;
22         this.color = color;
23     }
24
25     // ObjectId needs to be converted to string
26     public String get_id() { return _id.toHexString(); }
27     public void set_id(ObjectId _id) { this._id = _id; }
28
29     public String getName() { return name; }
30     public void setName(String name) { this.name = name; }
31
32     public String getSize() { return size; }
33     public void setSize(String size) { this.size = size; }
34
35     public String getColor() { return color; }
36     public void setColor(String color) { this.color = color; }
37
38 }
```

spring boot

Planets



spring

by Pivotal™

```
6 public class Planets {
7
8     @Id
9     public ObjectId _id;
10
11     public String name;
12     public String size;
13     public String color;
14
15     // Constructors
16     public Planets() {}
17
18     public Planets(ObjectId _id, String name, String size, String color) {
19         this._id = _id;
20         this.name = name;
21         this.size = size;
22         this.color = color;
23     }
24
25     // ObjectId needs to be converted to string
26     public String get_id() { return _id.toHexString(); }
27     public void set_id(ObjectId _id) { this._id = _id; }
28
29     public String getName() { return name; }
30     public void setName(String name) { this.name = name; }
31
32     public String getSize() { return size; }
33     public void setSize(String size) { this.size = size; }
34
35     public String getColor() { return color; }
36     public void setColor(String color) { this.color = color; }
37
38 }
```

spring boot

Planets



spring

by Pivotal™

```
6 public class Planets {
7
8     @Id
9     public ObjectId _id;
10
11     public String name;
12     public String size;
13     public String color;
14
15     // Constructors
16     public Planets() {}
17
18     public Planets(ObjectId _id, String name, String size, String color) {
19         this._id = _id;
20         this.name = name;
21         this.size = size;
22         this.color = color;
23     }
24
25     // ObjectId needs to be converted to string
26     public String get_id() { return _id.toHexString(); }
27     public void set_id(ObjectId _id) { this._id = _id; }
28
29     public String getName() { return name; }
30     public void setName(String name) { this.name = name; }
31
32     public String getSize() { return size; }
33     public void setSize(String size) { this.size = size; }
34
35     public String getColor() { return color; }
36     public void setColor(String color) { this.color = color; }
37
38 }
```


spring boot

Planets



spring

by Pivotal™

```
6 public class Planets {
7
8     @Id
9     public ObjectId _id;
10
11     public String name;
12     public String size;
13     public String color;
14
15     // Constructors
16     public Planets() {}
17
18     public Planets(ObjectId _id, String name, String size, String color) {
19         this._id = _id;
20         this.name = name;
21         this.size = size;
22         this.color = color;
23     }
24
25     // ObjectId needs to be converted to string
26     public String get_id() { return _id.toHexString(); }
27     public void set_id(ObjectId _id) { this._id = _id; }
28
29     public String getName() { return name; }
30     public void setName(String name) { this.name = name; }
31
32     public String getSize() { return size; }
33     public void setSize(String size) { this.size = size; }
34
35     public String getColor() { return color; }
36     public void setColor(String color) { this.color = color; }
37
38 }
```

spring boot



PlanetsController

```
10  @RestController
11  @RequestMapping("/")
12  public class PlanetsController {
13
14      @Autowired
15      private PlanetsRepository repository;
16
17      @RequestMapping("/planets")
18      @CrossOrigin(origins = "http://localhost:4200")
19      public List<Planets> getPlanets() { return repository.findAll(); }
20
21
22
23      @RequestMapping("/*")
24      public String greet() { return "Sorry no valid endpoint, try '/planets'!"; }
```


spring & spring boot

- vereinfachte Java Entwicklung
- Reduktion von boilerplate code, Annotations und XML Configuration
- Förderung best practices in Programmierung
- Dependency Management, Builds (Maven), etc.
- Microservices & Monolithen
- Spring ecosystem: JDBC, Data, Security



JHipster

- mächtiges boiler plate tool
- Projekt in einem Guss
- ...





Datenbank & Datenmodell



Datenbank



- sehr strikte Vorgaben (Datentypen)
- Tabellen können mit Joins verknüpft werden, in diesem Projekt aber nicht benötigt
- gibt es länger und daher auch größere Community
- flexibel
- kein Schema benötigt, bei Bedarf kann aber Mongoose verwendet werden
- gute Performance, wenn auch in diesem Umfang nicht bemerkbar

Fazit → MongoDB: Geringes Vorwissen aus WebDev 2 soll erweitert werden. Änderungen an den Collection zu Beginn ziemlich einfach.

Datenmodell

planets	orders	currencies	users
id	id	id	id
name_de	planet_de	code	email
name_en	planet_en	value	pw
total_size	user		
price_km2	size_km2		
sold_size_km2	date		
image	price		
available			

Beispiele für Datensätze in der Datenbank

1. Planeten (Collection planets):

```
> db.planets.findOne()
{
  "_id" : ObjectId("5ca9d0832a062373ca9b8008"),
  "name_de" : "Jupiter",
  "name_en" : "Jupiter",
  "total_size" : "61.420.000.000",
  "price_km2" : "1.000.000",
  "sold_size_km2" : "2",
  "image" : "../design_and_concept/images/jupiter01.jpg",
  "available" : "true"
}
```

2. Bestellungen (Collection orders):

```
> db.orders.findOne()
{
  "_id" : ObjectId("5ca9d14c2a062373ca9b8009"),
  "planet_de" : "Jupiter",
  "planet_en" : "Jupiter",
  "user" : "test123",
  "size_km2" : "2",
  "date" : "03.04.2019",
  "price" : "2.000.000"
}
```

3. Benutzer (Collection users):

```
> db.users.findOne()
{
  "_id" : ObjectId("5ca47515d46e3c4c5d7592a9"),
  "email" : "test@gmail.com",
  "pw" : "test123"
}
```

4. Währungen (Collection currencies):

```
> db.currencies.findOne()
{
  "_id" : ObjectId("5ca4a889867aaa42861ac651"),
  "code" : "EUR",
  "value" : "1"
}
```



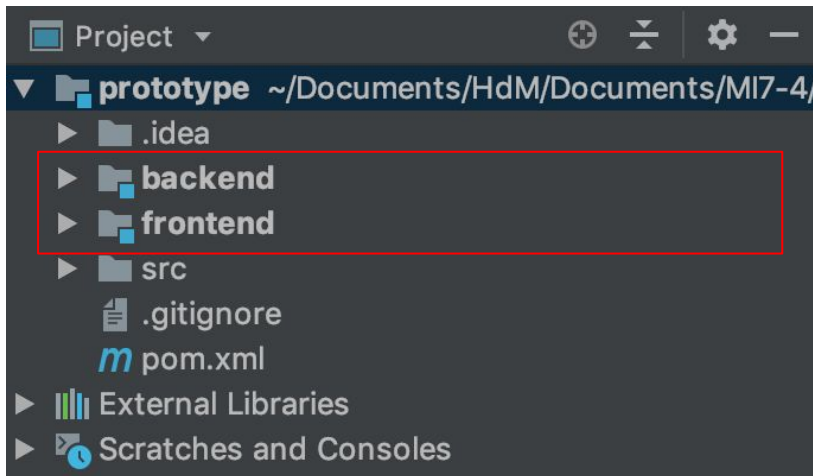
Projektstruktur & Installation



Projektstruktur

Saubere Trennung von front- & backend

Maven-Projekt mit 2 Modulen:



Installation Angular


Angular im Projektverzeichnis frontend installieren

```
npm install -g @angular/cli
```

Eine neue, leere App im aktuellen Verzeichnis erstellen

```
ng new client
```

Installation Spring Boot

 **Spring Initializr**
Bootstrap your application

Project

Language

Spring Boot

Project Metadata

Dependencies
[See all](#)

Maven Project

Gradle Project

Java

Kotlin

Groovy

2.2.0 M1

2.2.0 (SNAPSHOT)

2.1.5 (SNAPSHOT)

2.1.4

1.5.20

Group

com.astro

Artifact

space-estate

More options

Search dependencies to add

Web, Security, JPA, Actuator, Devtools...

Selected dependencies

Web [Web]
Servlet web application with Spring MVC and Tomcat

MongoDB [NoSQL]
Access MongoDB NoSQL Database with Spring Data MongoDB

.zip downloaden
und im backend
Verzeichnis entpacken.

Installation MongoDB



- Download für die entsprechende Plattform nötig - war bereits von WebDev 2 installiert

Problem: falscher Pfad für die Datenbank und Collections

Lösung: bei Start von mongod Dateipfad in Git Repository angeben

- danach Collections und Beispieldaten per Konsole einfügen



Prototype

Komponenten: Angular & Spring

planets-list.component.ts

```
export class PlanetsListComponent implements OnInit {  
  planets: Array<any>;  
  
  constructor(private planetService: PlanetService) { }  
  
  ngOnInit() {  
    this.planetService.getAllPlanets().subscribe( next: data => {  
      this.planets = data;  
    });  
  }  
}
```

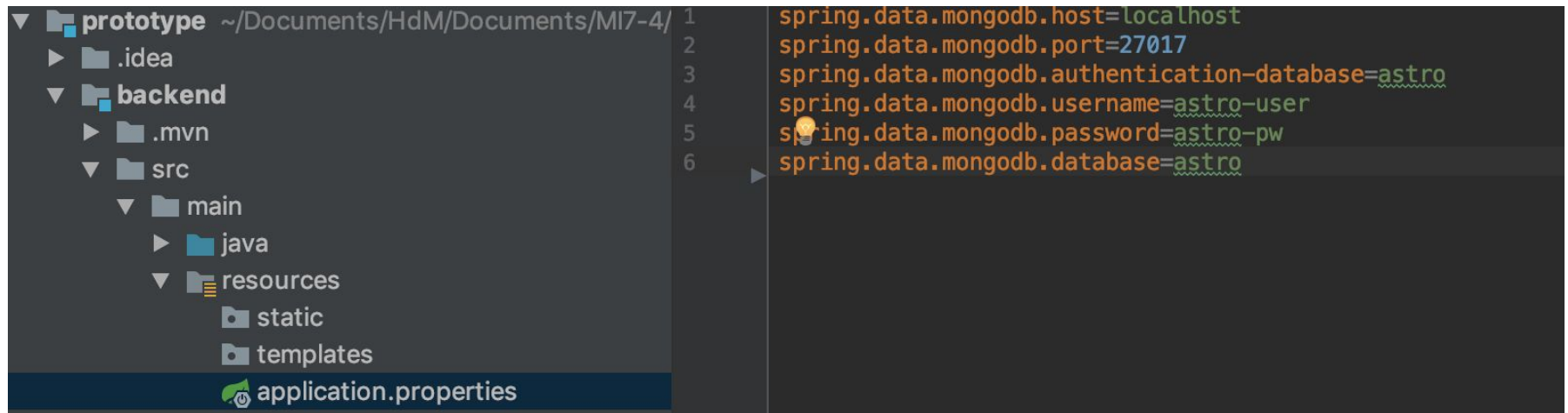
planets.service.ts

```
@Injectable()  
export class PlanetService {  
  constructor(private http: HttpClient) { }  
  
  getAllPlanets(): Observable<any> {  
    return this.http.get( url: 'localhost:8080/planets');  
  }  
}
```

planets-list.component.html

```
<h2>Planets List</h2>  
<ul>  
  <li *ngFor="let planet of planets">  
    <em>{{planet.name}}</em> | Size: {{planet.size}} | Color: {{planet.color}}  
  </li>  
</ul>
```

Komponenten: Spring & MongoDB



The screenshot shows an IDE with a project named 'prototype' located at '~/Documents/HdM/Documents/MI7-4/'. The project structure on the left includes a 'backend' folder with 'src/main/resources/application.properties' selected. The right pane shows the contents of this file, which are MongoDB connection properties.

```
1 spring.data.mongodb.host=localhost
2 spring.data.mongodb.port=27017
3 spring.data.mongodb.authentication-database=astro
4 spring.data.mongodb.username=astro-user
5 spring.data.mongodb.password=astro-pw
6 spring.data.mongodb.database=astro
```

PlanetsController.java



The screenshot shows the 'PlanetsController.java' file with the following code:

```
@RequestMapping("/planets")
@CrossOrigin(origins = "http://localhost:4200")
public List<Planets> getPlanets() { return repository.findAll(); }
```

A callout box on the right shows the MongoDB query used in the repository: `db.planets.find()`. An arrow points from the `findAll()` method call in the controller to this callout. The callout also includes the text 'RequestBody Pets'.