Spring Data
JPA / MongoDB

Michael Inden

Freiberuflicher Consultant und Trainer

Speaker Intro

- Michael Inden, Jahrgang 1971
- Diplom-Informatiker, C.v.O. Uni Oldenburg

- ~8 ¼ Jahre **SSE** bei Heidelberger Druckmaschinen AG in Kiel
- ~6 ¾ Jahre **TPL, SA** bei IVU Traffic Technologies AG in Aachen
- ~4 ¼ Jahre **LSA / Trainer** bei Zühlke Engineering AG in Zürich
- ~3 Jahre **TL / CTO** bei Direct Mail Informatics / ASMIQ in Zürich
- **Freiberuflicher Consultant, Trainer und Konferenz-Speaker**
- Autor und Gutachter beim dpunkt.verlag

E-Mail: michael.inden@hotmail.ch
Blog: https://jaxenter.de/author/minden
https://www.wearedevelopers.com/magazine/java-records

Kurse: Bitte sprecht mich an!

Agenda

Workshop Contents

- Introduction
- JPA Example
- Repositories
- MongoDB Example

PART 1:
Einführung

Spring Data

- provide a **familiar** and **consistent**, Spring-based programming model for **data access**
- makes it **easy** to use **relational** and **non-relational databases**, and **cloud-based data** services.
- **umbrella project** which contains **many subprojects** that are specific to a given database.

http://projects.spring.io/spring-data/

Spring Data Main Modules

- **Spring Data Commons** - Core Spring concepts underpinning every Spring Data project.
- **Spring Data JPA** - Makes it easy to implement JPA-based repositories.
- **Spring Data MongoDB** - Spring based, object-document support and repositories for MongoDB.
- …

Gleich und doch verschieden

JPA

```
@Entity
@Table(name="TUSR")
public class User {

  @Id
  private String id;

  @Column(name="fn")
  private String name;

  private Date lastLogin;

...
}
```

MongoDB

```
@Document(
collection="usr")
public class User {

  @Id
  private String id;

  @Field("fn")
  private String name;

  private Date lastLogin;

 ...
}
```

Neo4j

```
@NodeEntity
public class User {

  @GraphId
  Long id;


  private String name;

  private Date lastLogin;

...
}
```

JPA Example

Getting Started — Maven Dependencies

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.5.0</version>
</parent>


<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-rest</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
...
```

Getting Started — Gradle Dependencies

```
plugins {
  id "org.springframework.boot" version "2.5.0"
}

apply plugin: 'java'
apply plugin: 'eclipse'


repositories {
    mavenCentral()
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

dependencies {

    implementation 'org.springframework.boot:spring-boot-starter-data-jpa:2.5.0'
    testImplementation 'org.springframework.boot:spring-boot-starter-test:2.5.0'
```

First Spring Boot Application Example

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MyApp {

   public static void main(String[] args) {
      SpringApplication.run(MyApp.class, args);
   }
}
```

First Entity Example

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class SimpleEmployee
{
    @Id
    @GeneratedValue
    private Long id;
    private String firstName, lastName, description;

    private SimpleEmployee()
    {
    }

    public SimpleEmployee(String firstName, String lastName, String description)
    {
        this.firstName = firstName;
        this.lastName = lastName;
        this.description = description;
    }
```

...

First Repository Example

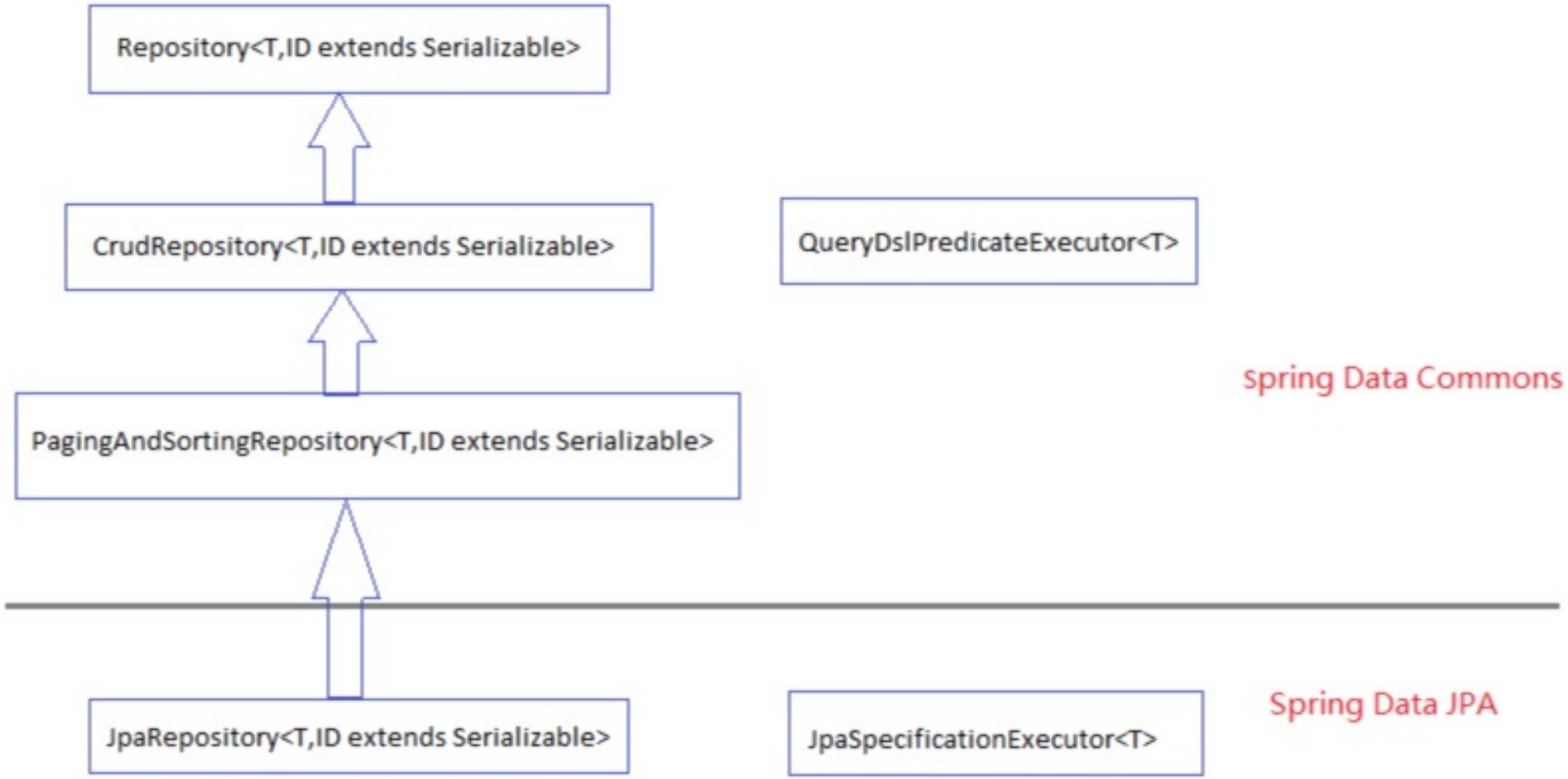
- Datenbankabfragen folgen dem **DAO Pattern**
- Diese sind durch sogenannte **Repositories** beschrieben
- In Spring sind das einfache **Interfaces (POJI)**
  => **Deklarative Programmierung**

```
import java.util.List;

import org.springframework.data.repository.CrudRepository;

public interface SimpleEmployeeRepository extends CrudRepository<SimpleEmployee, Long>
{
  SimpleEmployee findByFirstName(String firstName);

  List<SimpleEmployee> findByLastName(String lastName);
}
```

Basis Spring Repository — Standardfunktionalität

```
public interface CrudRepository<T, ID extends Serializable> extends Repository<T, ID> {

  <S extends T> S save(S entity);

  Optional<T> findById(ID primaryKey);

  Iterable<T> findAll();

  long count();

  void delete(T entity);

  boolean existsById(ID primaryKey);

  // …
}
```

Basis Spring Repositories



First Example

Starten wir einfach mal …

```
@SpringBootApplication
public class MyApp {

   public static void main(String[] args) {
      SpringApplication.run(MyApp.class, args);
   }
}
```

```
***************************
APPLICATION FAILED TO START
***************************

Description:

Cannot determine embedded database driver class for database type NONE
```

Wie binden wir eine DB ein?

H2 und Spring Boot

- Configuring the H2 database with Spring Boot is very easy: **Just add the H2 dependency to your POM**

- Spring Boot will automatically create the database, setup all the database JDBC objects, and by default configure Hibernate in a create-drop mode.

- Thus, when Hibernate starts up, it will scan the JPA annotated classes and automatically generate and execute the SQL code needed to create the database tables.

Additional H2 Dependencies

Maven

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
</dependency>
```

Gradle

```
implementation group: 'com.h2database', name: 'h2', version: '1.4.200'
```

Applikation starten

Maven

```
mvn clean package
mvn spring-boot:run
```

Gradle

```
gradle clean assemble
gradle bootRun
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.0)
```

Wie arbeiten wir mit der DB?

Datenbank befüllen — CommandLineRunner

```
@SpringBootApplication
public class Application implements CommandLineRunner
{
   @Autowired
   private SimpleEmployeeRepository repository;

   public static void main(String[] args)
   {
      SpringApplication.run(Application.class, args);
   }

   public void run(String... args) throws Exception
   {
      …
   }
}
```

Datenbank befüllen — CommandLineRunner

```
public void run(String... args) throws Exception
{
   Employee emp1 = new Employee("Michael", "Inden", "Team Lead");
   Employee emp2 = new Employee("Karthi", "Bollu Ganesh", "Lead Engineer");
   Employee emp3 = new Employee("Marcello", "Fluri", "Senior SW Engineer");

   System.out.println("Employees: " + repository.count());
   repository.save(emp1);
   repository.save(emp2);
   repository.save(emp3);
   System.out.println("Employees: " + repository.count());
   System.out.println("Employees: " + repository.findAll());

   // Find + Delete
   repository.delete(repository.findByFirstName("Marcello"));
   System.out.println("Employees: " + repository.count());
   System.out.println("Employees: " + repository.findAll());
}
```

Datenbank befüllen — CommandLineRunner

```
Employees: 0
Employees: 3
Employees: [SimpleEmployee [id=6, firstName=Michael, lastName=Inden, description=Team Lead],
SimpleEmployee [id=7, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer],
SimpleEmployee [id=8, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer]]
Employees: 2
Employees: [SimpleEmployee [id=6, firstName=Michael, lastName=Inden, description=Team Lead],
SimpleEmployee [id=7, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer]]
```

Spring Data Features

- Powerful repository and object-mapping abstractions
- Dynamic query derivation from repository method names
- Possibility to integrate custom repository code
- Implementation domain base classes providing basic properties
- Easy to combine with Spring REST controllers

Repositories

Abfrage-Varianten

- Methodennamen: findBy, readBy, getBy, countBy, queryBy
- GreaterThan, LessThan, Between
- Like, In
- Sortierung: OrderBy…Asc / Desc
- Eindeutigkeit: Distinct
- Einschränkungen / Paging: Top / First, etwa Top10

Schlüsselwörter

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Is,Equals	findByFirstname,findByFirstnameIs,findByFirstnameEquals	… where x.firstname = 1?
Between	findByStartDateBetween	… where x.startDate between 1? and ?2
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
IsNull	findByAgeIsNull	… where x.age is null
IsNotNull,NotNull	findByAge(Is)NotNull	… where x.age not null
Like	findByFirstnameLike	… where x.firstname like ?1
NotLike	findByFirstnameNotLike	… where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)
OrderBy	findByAgeOrderByLastnameDesc	… where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	… where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	… where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> age)	… where x.age not in ?1

Repository Example

```
public interface EmployeeRepository extends CrudRepository<Employee, Long>
{
   Employee findByFirstName(String firstName);
   List<Employee> findByLastName(String lastName);

   List<Employee> findByAgeGreaterThan(int age);
   List<Employee> findByAgeBetween(int lower, int upper);
   int countByAgeBetween(int lower, int upper);

   List<Employee> findTop3ByAgeLessThan(int maxAge);
   List<Employee> findByAgeLessThanOrderByFirstNameAsc(int maxAge);

   @Query("SELECT emp FROM Employee emp WHERE emp.firstName LIKE %?1%")
   List<Employee> getFirstNameLike(String firstName);

   List<Employee> findByFirstNameOrLastName(String firstName, String lastName);

   List<Employee> findByLastNameInAndAgeBetween(Collection<String> names,
                                                int lower, int upper);
}
```

Repository Example

```
Employee emp1 = new Employee("Michael", "Inden", "Team Lead", 47);
Employee emp2 = new Employee("Karthi", "Bollu Ganesh", "Lead Engineer", 33);
Employee emp3 = new Employee("Marcello", "Fluri", "Senior SW Engineer", 52);
Employee emp4 = new Employee("Marco", "Sonderegger", "SW Engineer", 30);
Employee emp5 = new Employee("Numa", "Trezzini", "SW Engineer", 30);
Employee emp6 = new Employee("Martin", "Dorta", "Senior SW Engineer", 50);

employeeRepository.save(emp1);
employeeRepository.save(emp2);
employeeRepository.save(emp3);
employeeRepository.save(emp4);
employeeRepository.save(emp5);
employeeRepository.save(emp6);
```

Repository Example

```
System.out.println("Employees 40-50: " + employeeRepository.findByAgeBetween(40, 50));
System.out.println("#Employees 40-50: " + employeeRepository.countByAgeBetween(40, 50));

System.out.println("Employees > 40: " + employeeRepository.findByAgeGreaterThan(40));
System.out.println("Employees < 50 Top 3: " + employeeRepository.findTop3ByAgeLessThan(50));
System.out.println("Employess: " + employeeRepository.findByAgeLessThanOrderByFirstNameAsc(35));
System.out.println("Employees: " + employeeRepository.getFirstNameLike("Ma"));
System.out.println("Employees: " + employeeRepository.findByFirstNameOrLastName("Michael",
                                                                                "Fluri"));
```

Repository Example

```
System.out.println("Employees 40-50: " + repository.findByAgeBetween(40, 50));
System.out.println("#Employees 40-50: " + repository.countByAgeBetween(40, 50));

System.out.println("Employees > 40: " + repository.findByAgeGreaterThan(40));
System.out.println("Employees < 50 Top 3: " + repository.findTop3ByAgeLessThan(50));
System.out.println("Employees: " + repository.findByAgeLessThanOrderByFirstNameAsc(35));

System.out.println("Employees: " + repository.getFirstNameLike("Ma"));
```

```
Employees 40-50: [Employee [id=1, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
                  Employee [id=6, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
#Employees 40-50: 2
Employees > 40: [Employee [id=1, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
                 Employee [id=3, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer, age=52],
                 Employee [id=6, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
Employees < 50 Top 3: [Employee [id=1, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
                       Employee [id=2, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer, age=33],
                       Employee [id=4, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30]]
Employess: [Employee [id=2, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer, age=33],
            Employee [id=4, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30],
            Employee [id=5, firstName=Numa, lastName=Trezzini, description=SW Engineer, age=30]]
Employees: [Employee [id=3, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer, age=52],
            Employee [id=4, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30],
            Employee [id=6, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
```

Repository Example

MongoDB Example

Repository Example

```
import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

@Document
public class Employee
{
   @Id
   private String id;

      …
}
```

Repository Example

```
public interface EmployeeRepository extends MongoRepository<Employee, String>
{
   Employee findByFirstName(String firstName);
   List<Employee> findByLastName(String lastName);

   List<Employee> findByAgeGreaterThan(int age);
   List<Employee> findByAgeBetween(int lower, int upper);
   int countByAgeBetween(int lower, int upper);

   List<Employee> findTop3ByAgeLessThan(int maxAge);
   List<Employee> findByAgeLessThanOrderByFirstNameAsc(int maxAge);

   List<Employee> getByFirstNameLike(String firstName);

   List<Employee> findByFirstNameOrLastName(String firstName, String lastName);

   List<Employee> findByLastNameInAndAgeBetween(Collection<String> names,
                                                int lower, int upper);
}
```

Repository Example

```
System.out.println("Employees 40-50: " + repository.findByAgeBetween(40, 50));
System.out.println("#Employees 40-50: " + repository.countByAgeBetween(40, 50));

System.out.println("Employees > 40: " + repository.findByAgeGreaterThan(40));
System.out.println("Employees < 50 Top 3: " + repository.findTop3ByAgeLessThan(50));
System.out.println("Employees: " + repository.findByAgeLessThanOrderByFirstNameAsc(35));

System.out.println("Employees: " + repository.getFirstNameLike("Ma"));
```

```
Employees 40-50: [Employee [id=5aa84d265131b00b822c12cc, firstName=Michael, lastName=Inden, description=Team Lead, age=47]]
#Employees < 50: 1
Employees > 40: [Employee [id=5aa84d265131b00b822c12cc, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
          Employee [id=5aa84d265131b00b822c12ce, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer, age=52],
          Employee [id=5aa84d265131b00b822c12d1, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
Employees < 50 Top 3:
         [Employee [id=5aa84d265131b00b822c12cc, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
          Employee [id=5aa84d265131b00b822c12cd, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer, age=33],
          Employee [id=5aa84d265131b00b822c12cf, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30]]
Employees:
      [Employee [id=5aa84d265131b00b822c12cd, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer, age=33],
       Employee [id=5aa84d265131b00b822c12cf, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30],
       Employee [id=5aa84d265131b00b822c12d0, firstName=Numa, lastName=Trezzini, description=SW Engineer, age=30]]
Employees:
      [Employee [id=5aa84d265131b00b822c12ce, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer, age=52],
       Employee [id=5aa84d265131b00b822c12cf, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30],
       Employee [id=5aa84d265131b00b822c12d1, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
Employees: [Employee [id=5aa84d265131b00b822c12cc, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
            Employee [id=5aa84d265131b00b822c12ce, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer, age=52]]
```

Repository Example

MongoDB: BETWEEN: lower < x < upper

```
Employees 40-50: [Employee [id=5aa84d265131b00b822c12cc, firstName=Michael, lastName=Inden, description=Team Lead, age=47]]
#Employees 40-50: 1
Employees > 40: [Employee [id=5aa84d265131b00b822c12cc, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
         Employee [id=5aa84d265131b00b822c12ce, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer, age=52],
         Employee [id=5aa84d265131b00b822c12d1, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
Employees < 50 Top 3:
        [Employee [id=5aa84d265131b00b822c12cc, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
         Employee [id=5aa84d265131b00b822c12cd, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer, age=33],
         Employee [id=5aa84d265131b00b822c12cf, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30]]
Employees:
      [Employee [id=5aa84d265131b00b822c12cd, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer, age=33],
       Employee [id=5aa84d265131b00b822c12cf, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30],
       Employee [id=5aa84d265131b00b822c12d0, firstName=Numa, lastName=Trezzini, description=SW Engineer, age=30]]
Employees:
      [Employee [id=5aa84d265131b00b822c12ce, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer, age=52],
       Employee [id=5aa84d265131b00b822c12cf, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30],
       Employee [id=5aa84d265131b00b822c12d1, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
```

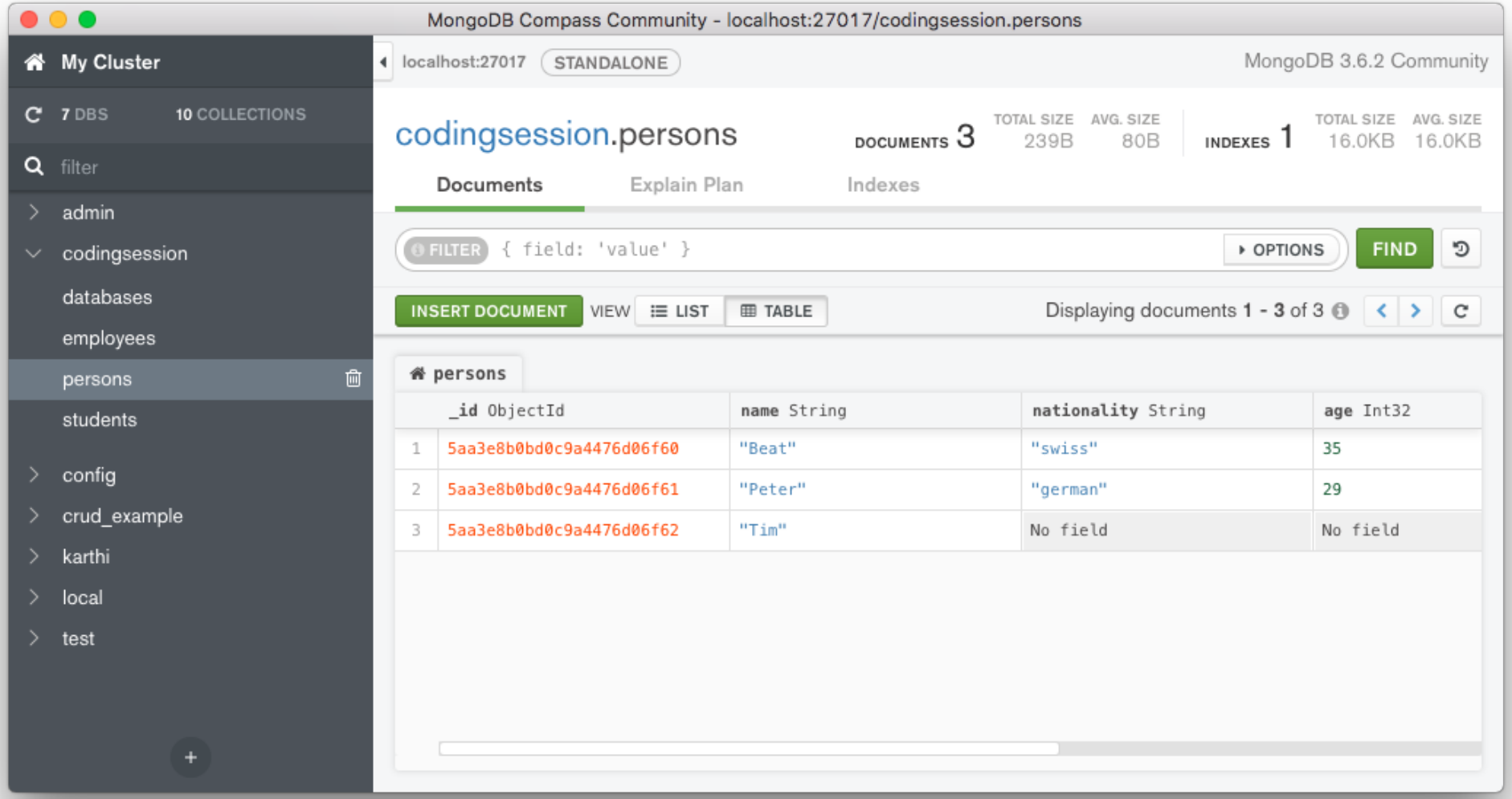
JPA: BETWEEN: lower <= x <= upper

```
Employees 40-50: [Employee [id=1, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
                  Employee [id=6, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
#Employees 40-50: 2
Employees > 40: [Employee [id=1, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
                 Employee [id=3, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer, age=52],
                 Employee [id=6, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
Employees < 50 Top 3: [Employee [id=1, firstName=Michael, lastName=Inden, description=Team Lead, age=47],
                       Employee [id=2, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer, age=33],
                       Employee [id=4, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30]]
Employees: [Employee [id=2, firstName=Karthi, lastName=Bollu Ganesh, description=Lead Engineer, age=33],
            Employee [id=4, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30],
            Employee [id=5, firstName=Numa, lastName=Trezzini, description=SW Engineer, age=30]]
Employees: [Employee [id=3, firstName=Marcello, lastName=Fluri, description=Senior SW Engineer, age=52],
            Employee [id=4, firstName=Marco, lastName=Sonderegger, description=SW Engineer, age=30],
            Employee [id=6, firstName=Martin, lastName=Dorta, description=Senior SW Engineer, age=50]]
```

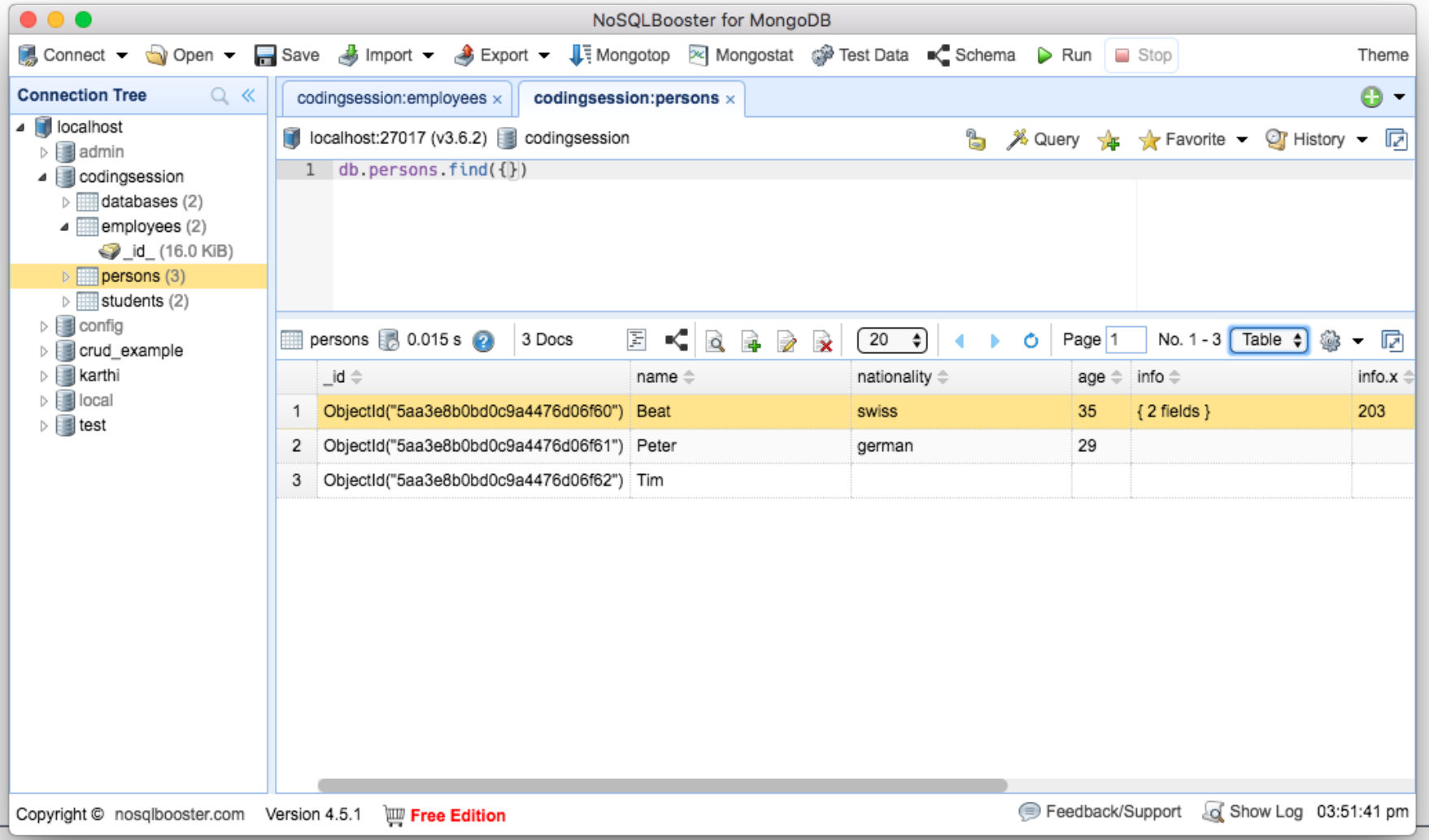
Repository Example

Names-Postfix	Operation als JSON
GreaterThan	`{ "age" : { "$gt" : <value> } }`
LessThan	`{ "age" : { "$lt" : <value> } }`
Between	`{ "age" : { "$gt" : from, "$lt" : to } }`
IsNotNull, NotNull	`{ "age" : { "$ne" : null } }`
IsNull, Null	`{ "age" : null }`
-/-	`{ "age" : <value> }`
Not	`{ "age" : { "$ne" : <value> } }`

MongoDB Compass



NoSQL Booster Query Tool



MongoDB Query Tools

- https://docs.mongodb.com/compass/master/install/
- https://nosqlbooster.com/downloads

Links zum Weitermachen

Zum weiteren Ausprobieren

- https://auth0.com/blog/integrating-spring-data-jpa-postgresql-liquibase/

- https://www.javaworld.com/article/2078898/open-source-tools/open-source-tools-open-source-java-projects-spring-data.html?page=2

Thank You