Kostenloses eBook

LERNEN spring-data

Free unaffiliated eBook created from
Stack Overflow contributors.

#spring-data

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: spring-data

It is an unofficial and free spring-data ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official spring-data.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Spring-Data

Bemerkungen

Moderne Softwareanwendungen haben die Möglichkeit, Daten in mehreren Datenspeichertypen zu speichern. Während traditionelle Datenspeicher wie relationale Datenbanken nach wie vor beliebt sind, sind NoSQL-Datenbanken und Cloud-basierter Speicher alltäglich geworden. Jeder dieser Datenspeichertypen hat seine eigenen Stärken und ist daher für verschiedene Arten von Geschäftsanwendungsfällen geeignet. Komplexe Geschäftsanwendungen verwenden daher mehr als einen Datenspeichertyp, um Datenspeicherungs-, Abruf- und Präsentationsvorgänge effizienter zu gestalten. Dies stellt die Herausforderung für Anwendungsprogrammierer dar, die sich mit der Komplexität des Verständnisses der von mehreren Datenspeichern bereitgestellten API und der angemessenen Verwendung dieser API in ihren Geschäftsanwendungen auseinandersetzen müssen.

Spring Data ist ein Projekt, das darauf abzielt, Anwendungsprogrammierern eine konsistente, benutzerfreundliche API bereitzustellen, unabhängig vom zugrunde liegenden Datenspeicher. Es kombiniert die Leistungsfähigkeit des Spring-Frameworks mit Konzepten bewährter Datenzugriffsparadigmen, wie z. B. domänengetriebenem Design , um Anwendungsprogrammierern eine vertraute und konsistente Grundlage für den Zugriff auf verschiedene Arten von Datenspeichern zu bieten, während die Besonderheiten eines zugrunde liegenden Datenspeichers erhalten bleiben gegebenenfalls.

Das Spring Data-Projekt besteht aus mehreren Unterprojekten, die als Bibliotheken für den Zugriff auf bestimmte Datenspeichertypen verwendet werden können. Alle von Spring Data unterstützten Datenspeicher und ihre Unterprojekte können auf der Hauptseite des Projekts abgerufen werden.

Examples

Installation oder Setup

Spring Data ist ein Projekt, das aus einer Reihe von Teilprojekten besteht. Die häufigsten sind Spring Data JPA , Spring Data MongoDB , Spring Data Elasticsearch , Spring Data Neo4J , Spring Data Cassandra und Spring Data Redis .

Wenn Sie nicht Ihr eigenes Teilprojekt auf der Grundlage von Spring Data entwickeln, ist es höchst unwahrscheinlich, dass Sie es direkt in Ihrer Anwendung verwenden müssen. Einzelheiten zu ihrer Installation und Einrichtung finden Sie in den einzelnen Unterprojekten. Wenn Sie jedoch Spring Data direkt in Ihrer Anwendung verwenden müssen, sind die folgenden Anweisungen hilfreich.

Maven benutzen

```
<dependencies>
  <dependency>
    <groupId>org.springframework.data</groupId>
```

```
    <artifactId>spring-data-commons</artifactId>
    <version>[version-number]</version>
  </dependency>
</dependencies>
```

Gradle verwenden

```
dependencies {
  compile 'org.springframework.data:spring-data-commons:[version-number]'
}
```

Ersetzen Sie *[Versionsnummer]* durch die Spring Data-Version, die Sie verwenden möchten.

Erste Schritte mit Spring-Data online lesen: https://riptutorial.com/de/spring-data/topic/5440/erste-schritte-mit-spring-data

Kapitel 2: Paginierung mit Frühlingsdaten

Einführung

Paginierung durch Übergeben von Parmeter mit benutzerdefinierter Abfrage in Spring Data JPA

Examples

Paginierung durch Übergeben von Parmeter mit benutzerdefinierter Abfrage in Spring Data JPA

Ich benutze Spring Boot 1.4.4.RELEASE, mit MySQL als Datenbank- und Spring Data JPA-Abstraktion, um mit MySQL zu arbeiten. In der Tat ist es das Spring Data JPA-Modul, das die Einrichtung von Pagination in einer Spring-Boot-App so einfach macht.

Szenario macht einen Endpunkt / Studenten / Klassenzimmer / {id} verfügbar. Es wird eine Liste der Schüler und andere Paging-Informationen zurückgegeben (die wir in einer Minute sehen würden), basierend auf den Seiten- und Größenparametern und der Klassenraum-ID, die mit übergeben wurden.

Zunächst erstelle ich eine Domain Student

```
@Entity
@Table(name = "student")
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

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

    @NotNull
    @Column(name = "rollnumber", nullable = false)
    private Integer rollnumber;

    @Column(name = "date_of_birth")
    private LocalDate dateOfBirth;

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

    @ManyToOne(optional = false)
    @NotNull
    private Classroom classroom;

//getter and setter

}
```

Schüler beziehen sich auf das Klassenzimmer

```
@Entity
@Table(name = "classroom")
public class Classroom  {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

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

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

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

//getter && setter

}
```

Wir haben RestController

```
@RestController
@RequestMapping("/api")
public class StudentResource {

private final StudentService studentService;

    public StudentResource(StudentService studentService) {
        this.studentService = studentService;
    }

 @GetMapping("/students/classroom/{id}")
    public ResponseEntity<Page<StudentDTO>> getAllStudentsBasedOnClassroom(@ApiParam Pageable
pageable,@PathVariable Long id)
        throws URISyntaxException {
        Page<StudentDTO> page = studentService.findByClassroomId(id, pageable);
        HttpHeaders headers = PaginationUtil.generatePaginationHttpHeaders(page,
"/api/students/classroom");
        return new ResponseEntity<Page<StudentDTO>>(page, headers, HttpStatus.OK);
    }

}
```

Beachten Sie, dass wir RequestParams nicht an unsere Handler-Methode übergeben haben. Wenn der Endpunkt / students / classroom / 1? Page = 0 & size = 3 getroffen wird, löst Spring die Parameter für Seite und Größe automatisch auf und erstellt eine Pageable-Instanz. Wir übergeben diese Pageable-Instanz dann an die Service-Schicht, die sie an unsere Repository-Schicht weitergibt.

Serviceklasse

```
public interface StudentService {
```

```
  Page<StudentDTO> findByClassroomId(Long id,Pageable pageable);

}
```

service impl (hier benutze ich StudentMapper, um Class über MapStruct in DTOby zu konvertieren oder wir können es manuell tun)

```
@Service
@Transactional
public class StudentServiceImpl implements StudentService{

 private final StudentRepository studentRepository;

    private final StudentMapper studentMapper;

    public StudentServiceImpl(StudentRepository studentRepository, StudentMapper
studentMapper) {
        this.studentRepository = studentRepository;
        this.studentMapper = studentMapper;
    }
@Override
    public Page<StudentDTO> findByClassroomId(Long id, Pageable pageable) {
         log.debug("Request to get Students based on classroom : {}", id);
         Page<Student> result = studentRepository.findByClassroomId(id, pageable);
         return result.map(student -> studentMapper.studentToStudentDTO(student));
    }

}
```

Dies ist eine Mapper-Schnittstelle

```
@Mapper(componentModel = "spring", uses = {})
public interface StudentMapper{

  StudentDTO   studentToStudentDTO(Student student);
}
```

In StudentRepository wurde dann eine benutzerdefinierte Methode erstellt

```
public interface StudentRepository extends JpaRepository<Student,Long> {

    Page<Student> findByClassroomId(Long id, Pageable pageable);

}
```

Dann erhalten Sie alle untenstehenden Informationen mit den jeweiligen Daten

```
"last": false,
  "totalElements": 20,
  "totalPages": 7,
  "size": 3,
  "number": 0,
  "sort": null,
  "first": true,
  "numberOfElements": 3
```

Paginierung mit Frühlingsdaten online lesen: https://riptutorial.com/de/spring-data/topic/9142/paginierung-mit-fruhlingsdaten

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Spring-Data	Community, manish, sunkuet02
2	Paginierung mit Frühlingsdaten	Aman Tuladhar, VISHWANATH N P