EBook Gratis

APRENDIZAJE spring-data

Free unaffiliated eBook created from
Stack Overflow contributors.

#spring-data

Tabla de contenido

Acerca de

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

Capítulo 1: Comenzando con los datos de primavera

Observaciones

Las aplicaciones de software modernas tienen la opción de almacenar datos en más de un tipo de almacén de datos. Mientras que los almacenes de datos tradicionales como las bases de datos relacionales siguen siendo populares, las bases de datos NoSQL y el almacenamiento basado en la nube también se han convertido en algo común. Cada uno de estos tipos de almacenes de datos tiene sus propias fortalezas y, por lo tanto, es adecuado para diferentes tipos de casos de uso empresarial. Por lo tanto, las aplicaciones empresariales complejas terminan utilizando más de un tipo de almacén de datos para hacer que las operaciones de almacenamiento, recuperación y presentación de datos sean más eficientes. Esto presenta el desafío de que los programadores de aplicaciones tengan que lidiar con la complejidad de comprender la API proporcionada por múltiples almacenes de datos y utilizar estas API de manera adecuada en sus aplicaciones empresariales.

Spring Data es un proyecto que tiene como objetivo proporcionar una API coherente y fácil de usar para los programadores de aplicaciones, independientemente del almacén de datos subyacente utilizado. Combina el poder del marco de Spring con conceptos de paradigmas de acceso a datos probados, como el diseño controlado por dominio, para proporcionar una base familiar y consistente a los programadores de aplicaciones para acceder a diferentes tipos de almacenes de datos, al tiempo que conserva las características específicas de un almacén de datos subyacente. donde corresponda.

El proyecto Spring Data consta de varios subproyectos que se pueden usar como bibliotecas para acceder a tipos específicos de almacenes de datos. El conjunto completo de almacenes de datos compatibles con Spring Data y sus subproyectos se puede obtener en la página principal del proyecto.

Examples

Instalación o configuración

Spring Data es un proyecto que consiste en una serie de subproyectos. Los más comunes son Spring Data JPA , Spring Data MongoDB , Spring Data Elasticsearch , Spring Data Neo4J , Spring Data Cassandra y Spring Data Redis .

A menos que esté desarrollando su propio subproyecto basado en Spring Data, es muy poco probable que necesite usarlo directamente en su aplicación. Consulte los subproyectos individuales para obtener detalles sobre su instalación y configuración. Sin embargo, si tiene la necesidad de usar Spring Data en su aplicación directamente, las siguientes instrucciones serán útiles.

Usando maven

```
<dependencies>
  <dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-commons</artifactId>
    <version>[version-number]</version>
  </dependency>
</dependencies>
```

Usando gradle

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

Sustituya *[número de versión]* con la versión de Spring Data que desea usar.

Lea Comenzando con los datos de primavera en línea: https://riptutorial.com/es/spring-data/topic/5440/comenzando-con-los-datos-de-primavera

Capítulo 2: Paginación con datos de primavera

Introducción

Paginación pasando parámetro con consulta personalizada en datos de primavera JPA

Examples

Paginación pasando parámetro con consulta personalizada en datos de primavera JPA

Uso Spring Boot 1.4.4.RELEASE, con MySQL como base de datos y Spring Data JPA abstraction para trabajar con MySQL. De hecho, es el módulo JPA de Spring Data lo que hace que sea tan fácil configurar Pagination en una aplicación de arranque Spring en primer lugar.

El escenario expone un punto final / estudiantes / aula / {id}. Devolverá una Lista de estudiantes y otra información de paginación (que veríamos en un minuto) según los parámetros de página y tamaño y el ID de clase de clase que se pasaron junto con ella.

Para empezar, creo un dominio 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

}
```

Estudiante relacionarse con aula

```
@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

}
```

tenemos 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);
    }

}
```

Tenga en cuenta que no hemos pasado RequestParams a nuestro método de controlador. Cuando se alcanza el punto final / estudiantes / aula / 1? Página = 0 y tamaño = 3, Spring resolvería automáticamente los parámetros de página y tamaño y crearía una instancia paginable. Luego pasaríamos esta instancia Pageable a la capa de Servicio, la cual lo pasaría a nuestra capa de Repositorio.

Clase de servicio

```
public interface StudentService {
```

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

}
```

service impl (aquí utilizo StudentMapper para convertir Class a DTOby usando mapStruct o podemos hacerlo manualmente)

```
@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));
    }

}
```

esta es la interfaz del mapeador

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

  StudentDTO   studentToStudentDTO(Student student);
}
```

A continuación, en StudentRepository i worte método personalizado

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

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

}
```

Luego nos dará toda la información a continuación con los datos respectivos.

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

Lea Paginación con datos de primavera en línea: https://riptutorial.com/es/spring-data/topic/9142/paginacion-con-datos-de-primavera

Creditos

S. No	Capítulos	Contributors
1	Comenzando con los datos de primavera	Community, manish, sunkuet02
2	Paginación con datos de primavera	Aman Tuladhar, VISHWANATH N P