eBook Gratuit

APPRENEZ spring-data

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#spring-data

Table des matières

À propos 1

Chapitre 1: Démarrer avec les données de printemps 2

Remarques 2

Examples 2

Installation ou configuration 2

Chapitre 2: Pagination avec données de printemps 4

Introduction 4

Examples 4

Pagination en passant le paramètre avec une requête personnalisée dans les données de prin 4

Crédits 8

À propos

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

Chapitre 1: Démarrer avec les données de printemps

Remarques

Les applications logicielles modernes ont la possibilité de stocker des données dans plusieurs types de magasin de données. Alors que les banques de données traditionnelles, telles que les bases de données relationnelles, restent populaires, les bases de données NoSQL et le stockage en nuage sont également devenus monnaie courante. Chacun de ces types de magasins de données possède ses propres atouts et convient donc à différents types de cas d'utilisation. Les applications métier complexes finissent donc par utiliser plusieurs types de magasin de données pour rendre les opérations de stockage, de récupération et de présentation des données plus efficaces. Ceci représente le défi des programmeurs d'applications devant gérer la complexité de comprendre l'API fournie par plusieurs magasins de données et d'utiliser ces API de manière appropriée dans leurs applications métier.

Spring Data est un projet qui vise à fournir une API cohérente et conviviale aux programmeurs d'applications, indépendamment du magasin de données sous-jacent utilisé. Il combine la puissance du framework Spring avec des concepts issus de paradigmes d'accès aux données éprouvés tels que la conception pilotée par le domaine pour fournir une base familière et cohérente aux programmeurs d'applications pour accéder à différents types de magasins de données, tout en conservant les spécificités d'un magasin de données sous-jacent. le cas échéant.

Le projet Spring Data se compose de plusieurs sous-projets pouvant être utilisés comme bibliothèques pour accéder à des types spécifiques de magasins de données. L'ensemble complet des magasins de données pris en charge par Spring Data et ses sous-projets peut être obtenu à partir de la page principale du projet.

Examples

Installation ou configuration

Spring Data est un projet composé de plusieurs sous-projets. Les plus courantes sont Spring Data JPA , Spring Data MongoDB , Spring Data Elasticsearch , Spring Data Neo4J , Spring Data Cassandra et Spring Data Redis .

Si vous ne développez pas votre propre sous-projet basé sur Spring Data, il est fort peu probable que vous deviez l'utiliser directement dans votre application. Voir les sous-projets individuels pour plus de détails sur leur installation et leur configuration. Si toutefois vous avez besoin d'utiliser Spring Data directement dans votre application, les instructions suivantes vous seront utiles.

Utiliser Maven

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

Utiliser Gradle

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

Remplacez *[numéro de version]* par la version Spring Data que vous souhaitez utiliser.

Lire Démarrer avec les données de printemps en ligne: https://riptutorial.com/fr/spring-data/topic/5440/demarrer-avec-les-donnees-de-printemps

Chapitre 2: Pagination avec données de printemps

Introduction

Pagination en passant le paramètre avec une requête personnalisée dans les données de printemps JPA

Examples

Pagination en passant le paramètre avec une requête personnalisée dans les données de printemps JPA

J'utilise Spring Boot 1.4.4.RELEASE, avec MySQL comme base de données et l'abstraction JPA de Spring Data pour travailler avec MySQL. En effet, c'est le module JPA de Spring Data qui facilite la mise en place de la pagination dans une application de démarrage Spring.

Le scénario expose un noeud final / students / classroom / {id}. Il renverra une liste d'étudiants et d'autres informations de pagination (que nous verrions en une minute) basées sur les paramètres de page et de taille et sur classroomId qui ont été transmis avec elle.

Pour commencer, je crée un domaine étudiant

```
@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
```

```
}
```

Étudiant en relation avec la classe

```
@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

}
```

nous avons 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);
    }

}
```

Notez que nous n'avons pas transmis RequestParams à notre méthode de gestionnaire. Lorsque le noeud final / students / classroom / 1? Page = 0 & size = 3 est sélectionné, Spring résout automatiquement les paramètres de page et de taille et crée une instance Pageable. Nous transmettions ensuite cette instance Pageable à la couche Service, qui la transmettrait à notre couche Référentiel.

Classe de service

```
public interface StudentService {

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

}
```

service impl (ici je utilisateur StudentMapper pour convertir Class en DTO en utilisant mapStruct ou nous pouvons faire manuellement)

```
@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));
    }

}
```

c'est l'interface du mappeur

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

  StudentDTO   studentToStudentDTO(Student student);
}
```

puis dans StudentRepository i worte méthode personnalisée

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

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

}
```

alors il nous donnera toutes les informations ci-dessous avec les données respectives

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

```
  "numberOfElements": 3
```

Lire Pagination avec données de printemps en ligne: https://riptutorial.com/fr/spring-data/topic/9142/pagination-avec-donnees-de-printemps

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec les données de printemps	Community, manish, sunkuet02
2	Pagination avec données de printemps	Aman Tuladhar, VISHWANATH N P