EBook Gratuito

APPRENDIMENTO spring-data

Free unaffiliated eBook created from
Stack Overflow contributors.

#spring-data

Sommario

Di

Capitolo 1: Iniziare con i dati di primavera

Osservazioni

Le moderne applicazioni software hanno la possibilità di memorizzare i dati in più di un tipo di archivio dati. Mentre i tradizionali archivi di dati come i database relazionali rimangono popolari, anche i database NoSQL e lo storage basato su cloud sono diventati comuni. Ciascuno di questi tipi di archivi di dati ha i suoi punti di forza ed è quindi adatto a diversi tipi di casi di utilizzo aziendale. Pertanto, le applicazioni aziendali complesse finiscono per utilizzare più di un tipo di archivio dati per rendere più efficienti le operazioni di archiviazione, recupero e presentazione dei dati. Questo presenta la sfida dei programmatori di applicazioni che devono affrontare la complessità della comprensione dell'API fornita da più archivi di dati e l'utilizzo di queste API in modo appropriato nelle loro applicazioni aziendali.

Spring Data è un progetto che mira a fornire un'API coerente e di facile utilizzo ai programmatori di applicazioni, indipendentemente dall'archivio dati sottostante utilizzato. Combina la potenza del framework Spring con concetti di comprovati paradigmi di accesso ai dati, come la progettazione basata sul dominio, per fornire una base familiare e coerente ai programmatori di applicazioni per accedere a diversi tipi di archivi dati, pur mantenendo le specifiche di un archivio dati sottostante, ove opportuno.

Il progetto Spring Data consiste in diversi sottoprogetti che possono essere utilizzati come librerie per accedere a tipi specifici di archivi dati. La serie completa di archivi dati supportati da Spring Data e dai relativi sottoprogetti può essere ottenuta dalla pagina principale del progetto.

Examples

Installazione o configurazione

Spring Data è un progetto costituito da un numero di sottoprogetti. I più comuni sono Spring Data JPA , Spring Data MongoDB , Spring Data Elasticsearch , Spring Data Neo4J , Spring Data Cassandra e Spring Data Redis .

A meno che non si stia sviluppando il proprio sottoprogetto basato su Spring Data, è altamente improbabile che sia necessario utilizzarlo direttamente nella propria applicazione. Vedere i singoli sottoprogetti per i dettagli sulla loro installazione e configurazione. Se tuttavia, è necessario utilizzare Spring Data nell'applicazione direttamente, le seguenti istruzioni saranno utili.

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]'
}
```

Sostituisci *[numero di versione]* con la versione di Spring Data che desideri utilizzare.

Leggi Iniziare con i dati di primavera online: https://riptutorial.com/it/spring-data/topic/5440/iniziare-con-i-dati-di-primavera

Capitolo 2: Impaginazione con dati primaverili

introduzione

Impaginazione passando parmetro con query personalizzata in dati primari JPA

Examples

Impaginazione passando parmetro con query personalizzata in dati primari JPA

Utilizzo Spring Boot 1.4.4.RELEASE, con MySQL come astrazione Database e Spring Data JPA per lavorare con MySQL. In effetti, è il modulo Spring Data JPA che semplifica la configurazione di Pagination in un'app di avvio Spring.

Lo scenario espone un endpoint / studenti / aula / {id}. Restituirà un elenco di studenti e altre informazioni di paging (che vedremmo in un minuto) in base alla pagina e ai parametri di dimensione e classeId che sono stati passati insieme ad esso.

Per cominciare, creo uno studente dominio

```
@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

}
```

Lo studente si relaziona con l'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

}
```

abbiamo 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);
    }

}
```

Si noti che non abbiamo passato RequestParams al nostro metodo di gestione. Quando viene colpito l'endpoint / studenti / classe / 1? Page = 0 e size = 3, Spring risolve automaticamente i parametri di pagina e dimensione e crea un'istanza Pageable. Passeremo quindi questa istanza Pageable al livello di servizio, che la trasferirà al nostro livello di repository.

Classe di servizio

```
public interface StudentService {

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

```
}
```

servizio impl (qui io utente StudentMapper per convertire Class to DTOby usando mapStruct o possiamo fare manualy)

```
@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));
    }

}
```

questa è l'interfaccia di mapper

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

  StudentDTO   studentToStudentDTO(Student student);
}
```

poi in StudentRepository ho usato un metodo personalizzato

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

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

}
```

quindi ci fornirà tutte le informazioni di seguito con i rispettivi dati

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

Leggi Impaginazione con dati primaverili online: https://riptutorial.com/it/spring-data/topic/9142/impaginazione-con-dati-primaverili

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con i dati di primavera	Community, manish, sunkuet02
2	Impaginazione con dati primaverili	Aman Tuladhar, VISHWANATH N P