

Spring Boot - Event Manager con validazioni

Obiettivo

Applicazione RESTful per gestione di eventi, partecipanti e location.

Struttura del progetto

```
event-manager/  
├── src/main/java/it/epicode/  
│   ├── config/  
│   │   └── FakerConfig.java  
│   ├── data/  
│   │   └── DataLoader.java  
│   ├── eventi/  
│   │   ├── Evento.java  
│   │   ├── EventoController.java  
│   │   ├── EventoRepository.java  
│   │   ├── EventoService.java  
│   │   ├── EventoRequest.java  
│   │   └── EventoResponse.java  
│   ├── location/  
│   │   ├── Location.java  
│   │   ├── LocationController.java  
│   │   ├── LocationRepository.java  
│   │   ├── LocationService.java  
│   │   ├── LocationRequest.java  
│   │   └── LocationResponse.java  
│   ├── partecipanti/  
│   │   └── Partecipante.java
```

```
| | | └─ PartecipanteController.java
| | | └─ PartecipanteRepository.java
| | | └─ PartecipanteService.java
| | | └─ PartecipanteRequest.java
| | | └─ PartecipanteResponse.java
| | └─ EventManagerApplication.java
└─ src/main/resources/
    └─ application.properties
└─ pom.xml
```

AGGIUNGI DIPENDENZE IN pom.xml

```
<dependencies>

    <!-- Faker -->
    <dependency>
        <groupId>com.github.javafaker</groupId>
        <artifactId>javafaker</artifactId>
        <version>1.0.2</version>
    </dependency>

    <!-- OpenAPI Swagger UI -->
    <dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
        <version>2.7.0</version>
    </dependency>
</dependencies>
```

Entità e Relazioni

Evento

- id: Long
- nome: String
- data: LocalDate
- location: ManyToOne Location
- partecipanti: OneToMany Partecipante

Partecipante

- id: Long
- nome, cognome, email, dataDiNascita
- evento: ManyToOne Evento

Location

- id: Long
- nome: String
- città: String
- eventi: OneToMany Evento

Evento.java

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Evento {
    @Id
```

```

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

@NotBlank
private String nome;

@Future
private LocalDate data;

@ManyToOne
@JoinColumn(name = "location_id")
private Location location;

@OneToMany(mappedBy = "evento")
private List<Partecipante> partecipanti;
}

```

Partecipante.java

```

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Partecipante {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank
    private String nome;

    @NotBlank
    private String cognome;
}

```

```
@Email
@NotBlank
private String email;

@Past
private LocalDate dataDiNascita;

@ManyToOne
@JoinColumn(name = "evento_id")
private Evento evento;
}
```

Location.java

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Location {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank
    private String nome;

    @NotBlank
    private String città;

    @OneToMany(mappedBy = "location")
```

```
    private List<Evento> eventi;  
}
```

DTO Request e Response

EventoRequest.java

```
@Data  
public class EventoRequest {  
    @NotBlank  
    private String nome;  
  
    @Future  
    private LocalDate data;  
  
    @NotNull  
    private Long locationId;  
}
```

EventoResponse.java

```
@Data  
public class EventoResponse {  
    private Long id;  
    private String nome;  
    private LocalDate data;  
    private String locationNome;  
    private String locationCittà;  
}
```

PartecipanteRequest.java

```
@Data
public class PartecipanteRequest {
    @NotBlank
    private String nome;

    @NotBlank
    private String cognome;

    @Email
    @NotBlank
    private String email;

    @Past
    private LocalDate dataDiNascita;

    @NotNull
    private Long eventoId;
}
```

PartecipanteResponse.java

```
@Data
public class PartecipanteResponse {
    private Long id;
    private String nome;
    private String cognome;
    private String email;
    private LocalDate dataDiNascita;
    private String eventoNome;
}
```

LocationRequest.java

```
@Data
public class LocationRequest {
    @NotBlank
    private String nome;

    @NotBlank
    private String città;
}
```

LocationResponse.java

```
@Data
public class LocationResponse {
    private Long id;
    private String nome;
    private String città;
}
```

EventoService.java

```
@Validated
@Service
public class EventoService {

    @Autowired
    private EventoRepository repository;

    @Autowired
    private LocationService locationService;
```



```

public EventoResponse create(@Valid EventoRequest request) {
    Evento entity = toEntity(request);
    repository.save(entity);
    return toResponse(entity);
}

public List<EventoResponse> getAll() {
    return repository.findAll().stream().map(this::toResponse).toList();
}

public Evento toEntity(EventoRequest request) {
    Evento entity = new Evento();
    BeanUtils.copyProperties(request, entity);
    entity.setLocation(locationService.getById(request.getLocation().getId()));
    return entity;
}

public EventoResponse toResponse(Evento entity) {
    EventoResponse response = new EventoResponse();
    BeanUtils.copyProperties(entity, response);
    response.setLocationNome(entity.getLocation().getNome());
    response.setLocationCittà(entity.getLocation().getCittà());
    return response;
}
}

```

PartecipanteService.java

```

@Service
@Validated
public class PartecipanteService {

```

```
@Autowired
private PartecipanteRepository repository;

@Autowired
private EventoService eventoService;

public PartecipanteResponse create(@Valid PartecipanteRequest request) {
    Partecipante entity = toEntity(request);
    repository.save(entity);
    return toResponse(entity);
}

public List<PartecipanteResponse> getAll() {
    return repository.findAll().stream().map(this::toResponse).toList();
}

public Partecipante toEntity(PartecipanteRequest request) {
    Partecipante entity = new Partecipante();
    BeanUtils.copyProperties(request, entity);
    entity.setEvento(eventoService.toEntity(request.getEvento()));
    return entity;
}

public PartecipanteResponse toResponse(Partecipante entity) {
    PartecipanteResponse response = new PartecipanteResponse();
    BeanUtils.copyProperties(entity, response);
    return response;
}
}
```

LocationService.java

```
@Service
@Validated
public class LocationService {

    @Autowired
    private LocationRepository repository;

    public LocationResponse create(@Valid LocationRequest request) {
        Location entity = new Location();
        BeanUtils.copyProperties(request, entity);
        Location saved = repository.save(entity);
        LocationResponse response = new LocationResponse();
        BeanUtils.copyProperties(saved, response);
        return response;
    }

    public List<LocationResponse> getAll() {
        return repository.findAll().stream().map(entity -> {
            LocationResponse response = new LocationResponse();
            BeanUtils.copyProperties(entity, response);
            return response;
        }).toList();
    }

    public Location getById(Long id) {
        return repository.findById(id).orElseThrow();
    }
}
```

FakerConfig.java

```
package it.epicode.config;
```

```
import com.github.javafaker.Faker;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.Locale;

@Configuration
public class FakerConfig {

    @Bean
    public Faker faker() {
        return new Faker(new Locale("it")); // Locale italiano
    }
}
```

DataLoader.java

```
@Component
public class DataLoader implements CommandLineRunner {

    @Autowired
    private EventoService eventoService;

    @Autowired
    private PartecipanteService partecipanteService;

    @Autowired
    private LocationService locationService;

    @Autowired
    private Faker faker;

    @Override
```

```
public void run(String... args) {  
    for (int i = 0; i < 3; i++) {  
        var locationReq = new LocationRequest();  
        locationReq.setNome(faker.company().name());  
        locationReq.setCittà(faker.address().city());  
        locationService.create(locationReq);  
    }  
  
    var locations = locationService.getAll();  
  
    for (int i = 0; i < 5; i++) {  
        var eventoReq = new EventoRequest();  
        eventoReq.setNome("Evento " + faker.book().title());  
        eventoReq.setData(LocalDate  
            .now()  
            .plusDays(faker.number().numberBetween(1, 30)));  
        eventoReq  
            .setLocation(locations.get(faker.random().nextInt(locations.size())));  
        eventoService.create(eventoReq);  
    }  
  
    var eventi = eventoService.getAll();  
  
    for (int i = 0; i < 20; i++) {  
        var pReq = new PartecipanteRequest();  
        pReq.setNome(faker.name().firstName());  
        pReq.setCognome(faker.name().lastName());  
        pReq.setEmail(faker.internet().emailAddress());  
        pReq.setDataDiNascita(LocalDate  
            .now()  
            .minusYears(faker.number().numberBetween(18, 50)));  
        pReq.setEvento(eventi.get(faker.random().nextInt(eventi.size())));  
        partecipanteService.create(pReq);  
    }  
}
```

```
}  
}
```

Controller

EventoController.java

```
@RestController  
@RequestMapping("/eventi")  
public class EventoController {  
  
    @Autowired  
    private EventoService service;  
  
    @GetMapping  
    public List<EventoResponse> getAll() {  
        return service.getAll();  
    }  
  
    @PostMapping  
    public ResponseEntity<EventoResponse> create(@RequestBody EventoRequest request) {  
        EventoResponse response = service.create(request);  
        return new ResponseEntity<>(response, HttpStatus.CREATED);  
    }  
}
```

PartecipanteController.java

```
@RestController  
@RequestMapping("/partecipanti")  
public class PartecipanteController {
```

```
@Autowired
private PartecipanteService service;

@GetMapping
public List<PartecipanteResponse> getAll() {
    return service.getAll();
}

@PostMapping
public ResponseEntity<PartecipanteResponse> create(@RequestBody PartecipanteRequest request) {
    PartecipanteResponse response = service.create(request);
    return new ResponseEntity<>(response, HttpStatus.CREATED);
}
}
```

LocationController.java

```
@RestController
@RequestMapping("/location")
public class LocationController {

    @Autowired
    private LocationService service;

    @GetMapping
    public List<LocationResponse> getAll() {
        return service.getAll();
    }

    @PostMapping
    public ResponseEntity<LocationResponse> create(@RequestBody LocationRequest request) {
        LocationResponse response = service.create(request);
        return new ResponseEntity<>(response, HttpStatus.CREATED);
    }
}
```

```
}  
}
```

Validazioni

Annotazione	Descrizione	Esempio
@DecimalMax	Valore massimo per un numero decimale	@DecimalMax("100.00") private BigDecimal sconto;
@DecimalMin	Valore minimo per un numero decimale	@DecimalMin("10.00") private BigDecimal prezzo;
@Digits	Definisce numero massimo di cifre intere e decimali	@Digits(integer=3, fraction=2) private BigDecimal costo;
@Negative	Valore numerico negativo	@Negative private int perdita;
@NegativeOrZero	Valore <= 0	@NegativeOrZero private int debito;
@Positive	Valore numerico positivo	@Positive private int guadagno;
@PositiveOrZero	Valore >= 0	@PositiveOrZero private int saldo;
@Pattern	Valore deve rispettare una regex	@Pattern(regex="^[A-Z]{3}[0-9]{3}\$") private String codice;
@NotNull	Il valore non può essere null	@NotNull private Long id;
@NotBlank	Il campo non può essere vuoto o solo spazi	@NotBlank private String nome;
@NotEmpty	Il campo non può essere una stringa vuota	@NotEmpty private String codice;
@Size	Definisce la lunghezza minima e massima	@Size(min=3, max=50)
@Min	Il numero deve essere >= al valore specificato	@Min(18) private int età;

Annotazione	Descrizione	Esempio
@Max	Il numero deve essere <= al valore specificato	@Max(65) private int età;
@Email	Il valore deve avere un formato email valido	@Email private String email;
@Past	La data deve essere nel passato	@Past private LocalDate nascita;
@PastOrPresent	La data può essere passata o odierna	@PastOrPresent private LocalDate data;
@Future	La data deve essere nel futuro	@Future private LocalDate evento;
@FutureOrPresent	La data può essere futura o odierna	@FutureOrPresent private LocalDate e;