

Method-level Security in Spring

Introduction

Method-level Security in Spring

Method-level security in Spring allows you to secure individual methods in your application, providing fine-grained access control.

Method-level Security in Spring

What it is:

- It's a feature that lets you add security checks directly to individual methods,
- You use special annotations to tell Spring which methods need protection,
- Authorization checks.

How it's implemented:

- Main annotations are: `@PreAuthorize`, `@PostAuthorize`,
- You put these annotations right above the method you want to secure.

Example

Spring Security Annotation

Spring Security Expression Language(SpEL)

```
@PreAuthorize("authentication.principal.equals(#userId.toString())")
```

```
@GetMapping("/{userId}")
```

```
public Mono<ResponseEntity<UserRest>> getUser(@PathVariable("userId") UUID userId) {
```

```
    return userService
```

```
        .getUserById(userId)
```

```
        .map(userRest -> ResponseEntity
```

```
            .status(HttpStatus.OK)
```

```
            .body(userRest)
```

```
        );
```

```
}
```

Example

Spring Security Annotation

Spring Security Expression Language(SpEL)

```
@PostAuthorize("returnObject.body.owner == authentication.principal")
```

```
@GetMapping("/{id}")
```

```
public Mono<ResponseEntity<Document>> getDocument(@PathVariable Long id) {
```

```
    return documentService.findById(id)
```

```
        .map(ResponseEntity::ok)
```

```
        .switchIfEmpty(Mono.just(ResponseEntity.notFound().build()));
```

```
}
```