Authorization in Spring Security

Permissions, roles and beyond

Daniel Garnier-Moiroux

Spring I/O, 2025-05-22

Daniel Garnier-Moiroux

Software Engineer @ Broadcom

- Spring
- @garnier.wf
- https://garnier.wf/
- github.com/Kehrlann/
- contact@garnier.wf

Spring Security: Architecture Principles

Spring I/O 2024! On youtube!

Spring Security Authorization

1. 🌿 A demo app
2. 🛠️ Spring Security's authz tooling
    1. Request-level
    2. Method-level
    3. Object-level
3. 🧠 Authorization design
    1. Thinking about authorization
    2. Information is key

Spring Security Authorization

1. 🌿 **A demo app**
2. 🛠️ Spring Security's authz tooling
    1. Request-level
    2. Method-level
    3. Object-level
3. 🧠 Authorization design
    1. Thinking about authorization
    2. Information is key

A demo app

You've done this before! Either request-level or method-level security.

Separation of concerns: avoid security-related code in your domain code

Spring Security Authorization

1. 🌿 A demo app
2. 🛠️ **Spring Security's authz tooling**
    1. **Request-level**
    2. Method-level
    3. Object-level
3. 🧠 Authorization design
    1. Thinking about authorization
    2. Information is key

Request-level authorizarion

```
http.authorizeHttpRequests(auth → { /* ... */ });
```

Simple rules:

- `.permitAll()`, `.denyAll()`, `.authenticated()`,...
- `.hasRole()`, `.hasAuthority()`,...

Request-level authorizarion

For more interesting rules:

```
.access((authSupplier, reqContext) → { /* ... */ });
```

Composability:

```
.access(AuthorizationManagers.anyOf( ... ))
```

⚠ Does not read the body of the request

Spring Security Authorization

1. 🌿 A demo app
2. 🛠️ **Spring Security's authz tooling**
    1. Request-level
    2. **Method-level**
    3. Object-level
3. 🧠 Authorization design
    1. Thinking about authorization
    2. Information is key

Method-level authorizarion

`@PreAuthorize( ... )`, `@PostAuthorize( ... )`, `@PostFilter( ... )` with SpEL expressions.

Avoid complex expressions, and use a bean reference:
`@PreAuthorize("@authzService.authorize( ... )")`

Consider custom annotations for de-duplication:
`@AllowedDomains(domains = { "corp.example.com" })`

Method-level authorizarion

Even with `@PostAuthorize( ... )` or `@PostFilter( ... )`
DO filter in your database / service, e.g.:
`SELECT ... WHERE owner.id == authenticationId`

Enforce separation of concerns with
`@HandleAuthorizationDenied(handlerClass = ... )`

Spring Security Authorization

1. 🌿 A demo app
2. 🛠️ **Spring Security's authz tooling**
    1. Request-level
    2. Method-level
    3. **Object-level**
3. 🧠 Authorization design
    1. Thinking about authorization
    2. Information is key

Object-level authorization

You can apply security to object methods.

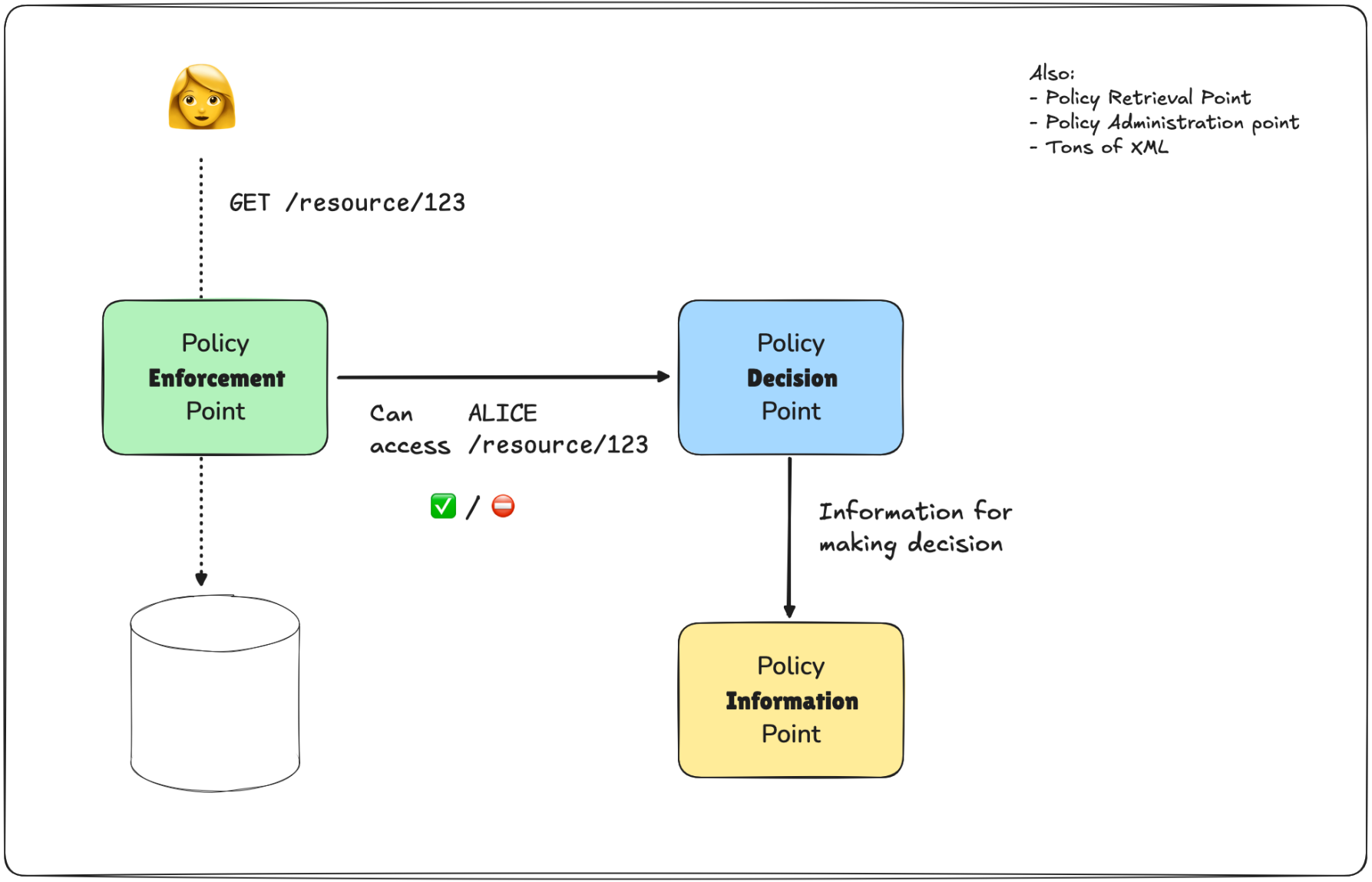
Annotate call site with `@AuthorizeReturnObject` to create a proxy and enforce those methods.

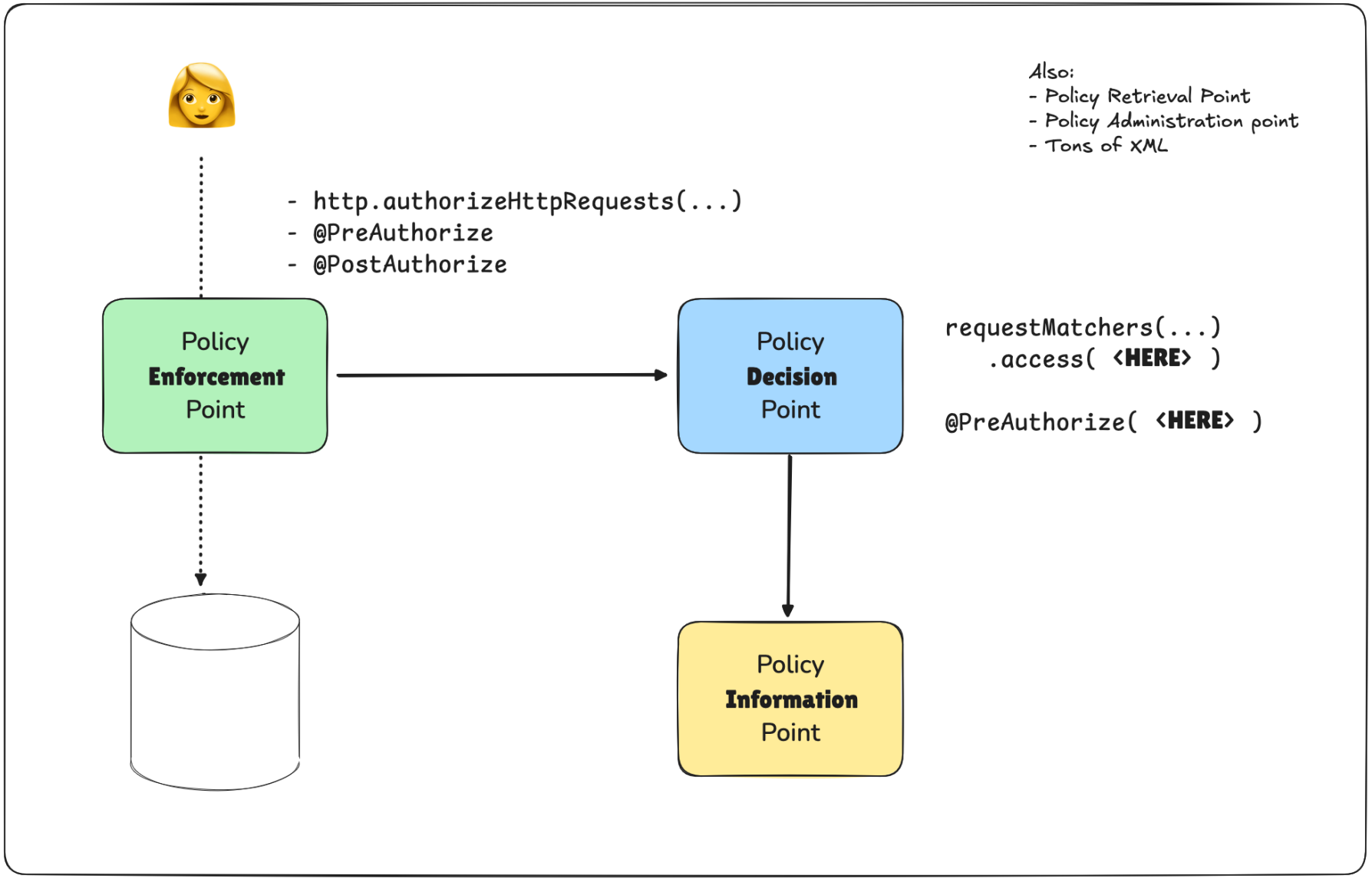
Spring Security Authorization

1. 🌿 A demo app
2. 🛠️ Spring Security's authz tooling
    1. Request-level
    2. Method-level
    3. Object-level
3. 🧠 **Authorization design**
    1. **Thinking about authorization**
    2. Information is key

Thinking about authorization

Remember XACML?


Also:
- Policy Retrieval Point
- Policy Administration point
- Tons of XML
GET /resource/123
Policy
Enforcement
Point
Can ALICE
access /resource/123
✅ / ⛔
Policy
Decision
Point
Information for
making decision
Policy
Information
Point


Also:
- Policy Retrieval Point
- Policy Administration point
- Tons of XML
- http.authorizeHttpRequests(...)
- @PreAuthorize
- @PostAuthorize
Policy Enforcement Point
Policy Decision Point
Policy Information Point
requestMatchers(...)
.access(<HERE>)
@PreAuthorize(<HERE>)

Spring Security Authorization

1. 🌿 A demo app
2. 🛠️ Spring Security's authz tooling
    1. Request-level
    2. Method-level
    3. Object-level
3. 🧠 **Authorization design**
    1. Thinking about authorization
    2. **Information is key**

Information is key

- What information is relevant for the security decision?
- Where can you capture or transform it?
- Consider:
  - `AuthenticationConverter` works on HttpServletRequest
  - `AuthenticationProvider` works on Authentication
  - `AuthenticationDetailsSource` also HttpServletRequest
  - Last resort: `Filter`

Information is key

Context matters!

Dedicated `SecurityFilterChain`s with different rules bring the complexity down:

- Simpler information gathering
- Simpler rules

PSA: Don't create 65 `SecurityFilterChain`s.

References

https://github.com/Kehrlann/spring-security-authorization

- @garnier.wf
- https://garnier.wf/
- contact@garnier.wf


Merci 😊
VMware Tanzu
VOXXED DAYS
ZURICH