Table of Contents

1. Security Annotations: FunctionalMapping and FunctionalAction

Quantum introduces two annotations to simplify declaring the functional context of REST resources and methods. The SecurityFilter reads these annotations to populate ResourceContext for downstream rule evaluation.

1.1. @FunctionalMapping (class-level)

```
import com.e2eq.framework.annotations.FunctionalMapping;

@FunctionalMapping(area = "sales", domain = "order")
@Path("/annotated")
public class AnnotatedTestResource {
    // ... resource endpoints ...
}
```

- Declares the Functional Area and Functional Domain for the resource class.
- In SecurityFilter, when a request targets a method on this class, the ResourceContext area/domain are taken from this annotation.

1.2. @FunctionalAction (method-level)

```
import com.e2eq.framework.annotations.FunctionalAction;

@POST
@Path("/create")
@FunctionalAction("CREATE")
public Response create(...) { ... }
```

- Declares the functional action for this endpoint (CREATE, VIEW, UPDATE, DELETE, etc.).
- If absent, SecurityFilter infers the action from the HTTP method:
- GET → VIEW
- POST → CREATE
- PUT/PATCH → UPDATE
- DELETE → DELETE

1.3. How SecurityFilter uses the annotations

At request time, SecurityFilter examines the matched resource class and method:

1. If @FunctionalMapping is present on the class, it sets area/domain accordingly.
2. If @FunctionalAction is present on the method, it uses that action; otherwise it infers from

HTTP method.

3. It then constructs and stores a ResourceContext(area, domain, action) into a thread-local SecurityContext for use by RuleContext and repositories.
4. The filter clears contexts after the response to avoid leaks across requests.

1.4. Example

The test resource and tests illustrate expected behavior:

- GET /annotated/view
- Class annotated with @FunctionalMapping(area="sales", domain="order")
- Method annotated with @FunctionalAction("VIEW")
- Response body from the sample shows: "sales:order:VIEW"
- POST /annotated/create
- Class annotated with @FunctionalMapping(area="sales", domain="order")
- Method NOT annotated with @FunctionalAction
- SecurityFilter infers action CREATE from HTTP method POST
- Response body shows: "sales:order:CREATE"

1.5. Writing Rules that target annotated resources

With ResourceContext reliably set, you can author rules that match on:

- area == "sales"
- functionalDomain == "order"
- action == "VIEW" | "CREATE" | "UPDATE" | "DELETE"

Rules can then attach filters via andFilterString/orFilterString that will apply to repository queries executed within these endpoints.