Spring.rest

InformAtelier

REST

Representational State Transfer (REST) is a type of software architecture for distributed systems.

InformAtelier

REST abstract

- An important concept in REST is the existence of resources (sources of information), which can be accessed through a global identifier (a URI). To use resources, the components of a network (client and server components) communicate through a standard interface (eg HTTP) and exchange representations of these resources (the document that transmits the information).

Verbi HTTP

- GET
- POST, PATCH,
- PUT, DELETE, HEAD
- OPTIONS
- TRACE

InformAtelier

URL Params

```
http://localhost:8080/sitename/controllerLevelMapping/1?someAttr=6
```

```
@RequestMapping("/{someID}")
public @ResponseBody int getAttr(@PathVariable(value="someID")
                                                                  String id,
                                        @RequestParam String someAttr) {
}
```

InformAtelier

@Request-ResponseBody

```
@Controller
@RequestMapping("/mail")
public class ExampleMailController {

   @PostMapping("/credentials")
   @ResponseBody
   public ResponseCredentials postCredentials(
        @RequestBody LoginForm loginForm) {
             return new ResponseCredentials("Thanks For Posting!!!");
        }
}

// {"text":"Thanks For Posting!!!"}
```

InformAtelier

ResponseEntity

```
@GetMapping("/hello")
ResponseEntity<String> hello() {
    return ResponseEntity.ok("Hello World!");
}
@GetMapping("/hello2")
ResponseEntity<String> hello2() {
    return ResponseEntity.status(HttpStatus.OK)
        .header("Custom-Header", "foo")
        .body("Custom header set");
}
```

@RestController

```
@RestController
@RequestMapping("books-rest")
public class SimpleBookRestController {

    @GetMapping("/{id}", produces = "application/json")
    public Book getBook(@PathVariable int id) {
        return findBookById(id);
    }
}
// @ResponseBody isn’t required
```

Actuator

- Actuator is mainly used to expose operational information about the running application – health, metrics, info, dump, env, etc. It uses HTTP endpoints or JMX beans to enable us to interact with it.
- Actuator comes with most endpoints disabled, only two are available by default: /health and /info.

Actuator Endpoints (I)

- /auditevents – lists security audit-related events such as user login/logout. Also, we can filter by principal or type among others fields
- /beans – returns all available beans in our BeanFactory. Unlike /auditevents, it doesn’t support filtering
- /conditions – formerly known as /autoconfig, builds a report of conditions around auto-configuration
- /configprops – allows us to fetch all @ConfigurationProperties beans
- /env – returns the current environment properties. Additionally, we can retrieve single properties
- /flyway – provides details about our Flyway database migrations
- /health – summarises the health status of our application
- /heapdump – builds and returns a heap dump from the JVM used by our application
- /info – returns general information. It might be custom data, build information or details about the latest commit

Actuator Endpoints (II)

- /liquibase – behaves like /flyway but for Liquibase
- /logfile – returns ordinary application logs
- /loggers – enables us to query and modify the logging level of our application
- /metrics – details metrics of our application. This might include generic metrics as well as custom ones
- /prometheus – returns metrics like the previous one, but formatted to work with a Prometheus server
- /scheduledtasks – provides details about every scheduled task within our application
- /sessions – lists HTTP sessions given we are using Spring Session
- /shutdown – performs a graceful shutdown of the application
- /threaddump – dumps the thread information of the underlying JVM

Richardson Maturity Model

- Level0: HTTP tunneling, single RPC endpoint
- Level1: Individual Resources
- Level2: HTTP verbs and response
- Level3: Hypermedia Links

HEATOAS

- Hypermedia as the Engine of Application State
- A hypermedia-driven site provides information to navigate the site's REST interfaces dynamically by including hypermedia links with the responses.

HEATOAS Example

```
{
  "name": "Alice",
  "links": [ {
    "rel": "self",
    "href": "http://localhost:8080/customer/1"
  } ]
}
```

InformAtelier

Guide

- https://spring.io/guides/gs/spring-boot/
- https://spring.io/guides/gs/rest-hateoas/

Riferimenti

- http://restcookbook.com/
- https://docs.spring.io/spring-hateoas/docs/current/reference/html/
- https://martinfowler.com/articles/richardsonMaturityModel.html
- https://www.iana.org/assignments/link-relations/link-relations.xhtml
- https://spring.io/understanding/HATEOAS