Designing RESTful APIs

https://github.com/essentialprogramming/undertow-spring-swagger

Outline

- Introduction to HTTP
- Representational State Transfer (REST)
- Examples of RESTful APIs
- Designing an API

Introduction to HTTP

Internet vs. World Wide Web

What is the difference between Internet and World Wide Web?

Internet is the network of computers.

World Wide Web is an application on top of internet.
(Like many others including email, ftp, telnet, ssh etc.)

World Wide Web is the killer app of the internet.

It revolutioned the internet.

World Wide Web - Key Concepts

- Uniform Resource Locator (URL)
- Hyper Text Markup Language (HTML)
- Hyper Text Transfer Protocol (HTTP)

Uniform Resource Location

Locate any resource with a single string.

Examples:

http://www.example.com/path/to/name

https://romania.voxxeddays.com/2020/03/01/api-design-strategy-the-ultimate-survival-guide/

Revolutionary idea!

Hyper Text

Document with references to other documents, which can be accessed immediately.

```
The term hypertext is coined by
<a href="https://en.wikipedia.org/wiki/Ted_Nelson">Ted Nelson</a>
in 1963.
```

Very simple idea. Nothing comes closer even after half a century.

Think: how do you manage related word documents?

Hyper Text Transfer Protocol (HTTP)

HTTP is the protocol to transfer hypertext.

Simple text-based protocol.

HTTP - Sample Request

```
GET /hello?name=web HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0
Accept: */*
```

HTTP - Sample Response

```
HTTP/1.1 200 OK
Server: gunicorn/19.7.1
Date: Thu, 11 May 2017 10:46:00 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 11

Hello, web!
```

HTTP - Important Parts

- HTTP Methods
  - GET, POST, PUT, DELETE
- Headers
  - Content-Type, Content-length, ...
- Status Codes
  - 200 OK, 404 Not Found

Safe and Idempotent Methods

- Safe - no side effects
  - GET and HEAD
- idempotence - the side-effects of more than one identical requests is the same as for a single request.
  - GET, HEAD, PUT and DELETE

Representational State Transfer (REST)

What is REST?

Architectural principles and constraints for building network-based application software.

Defined by *Roy Fielding* in his PhD dissertation "Architectural Styles and the Design of Network-based Software Architectures"

REST architectural pattern is based around two basic principles:

Resources as URLs: A resource is something like an entity or a noun in modelling lingo. Anything on a web is identified as a resource and each unique resource is identified by a unique URL.

Operations as HTTP methods: REST leverages existing HTTP methods, particularly GET, PUT, POST, and DELETE which map to resource's read, create, modify and removal operations respectively.

Any action performed by a client over HTTP, contains an URL and a HTTP method. The URL represents the resource and the HTTP method represents the action which needs to be performed over the resource.

Constraints in REST

- Give everything its own URI
- Use a Uniform Interface
    - GET, PUT, DELETE, POST
- Use hypermedia
    - Link your resources together
- Avoid session state
- Support Caching
- Communicate Representations
    - Support different MIME Media Types

Practical REST

- Thinking in Resources
  - model your application around resources/topics (nouns) instead of actions (verbs)
- Use HTTP methods and headers for metadata and control data

Practical REST - Resources

BAD

```
/show-page?id=5
/add-comment?post_id=5
```

GOOD

```
/pages/5
/pages/5/comments
```

Practical REST - HTTP Methods

Use HTTP methods for verbs. Common CRUD operations can be mapped to standard HTTP methods.

GET - read
POST - create
PUT - create or update
DELETE - delete

Practical REST - Resources

Golden rules of REST

- „We only need two URLs“
- „Verbs are bad, nouns are good“
- „Plurals are even better“
- „The web is your friend“
- „There is always a root (associations)“
- „There is always a parameter (complex stuff)“

Practical REST - Resources

„There is always a root“

- GET /orders/123/ingredients
- GET /orders/123/ingredients/333
- GET /orders/123/ingredients/milk
- PUT /orders/123/ingredients/milk
- POST /orders/123/ingredients/milk
- DELETE /orders/123/ingredients/milk

Practical REST - HTTP Status Codes

Use HTTP Status codes to indicate success and error cases.

SUCCESS

200 OK - Success

201 Created - New resouce is created successfully.

CLIENT ERRORS

400 Bad Request - malformed syntax
401 Unauthorized - authorization required
403 Forbidden - the current user doesn't have permission to access this resource
404 Not Found - requested resource is not found

SERVER ERRORS

500 Internal Error - Oops! something went wrong

501 Not Implemented - Not yet implemented!

Practical REST - HTTP Headers

Sample Request Headers

Accept: application/json
Accept-Language: te, en;q=0.9, kn;q=0.5
Authorization: Basic dGVzdDp0ZXN0

Sample Response Headers

Content-Type: application/json
Content-Language: en

Alternatives to REST

- SOAP
- XML-RPC
- HTTP-RPC (even with JSON)

HTTP RPC

```
$ curl -i https://slack.com/api/api.test
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
...

{"ok":true}
```

Good Examples of RESTful APIs

Github
https://developers.github.com/

Stripe
https://stripe.com/docs/api

Bad Examples of RESTful APIs

Flickr

https://www.flickr.com/services/api/

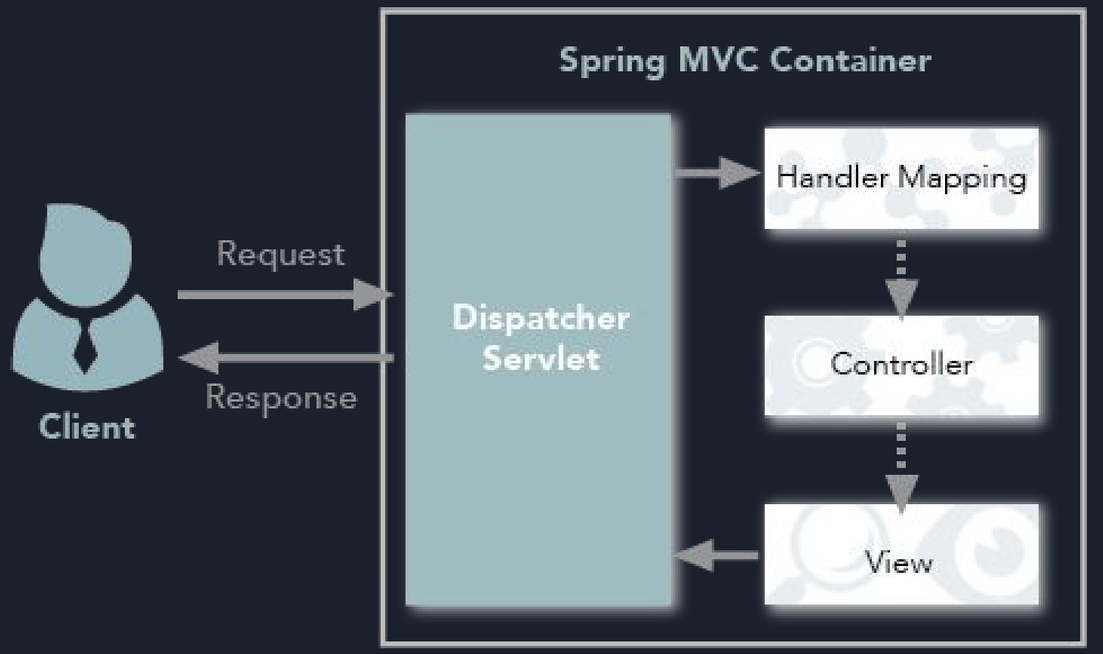
Bitly

http://dev.bitly.com/links.html

Spring MVC

Traditional Model View Controller

- The Model encapsulates the application data and in general, they will consist of POJO.
- The View is responsible for rendering the model data and in general, it generates HTML output that the client's browser can interpret.
- The Controller is responsible for processing User Requests and Building Appropriate Model and passes it to the view for rendering.



https://www.tutorialspoint.com/springmvc/springmvc_overview.htm

Controller

- **The Controller is the central piece in the Web Tier of the RESTful API**
- **It receives HTTP requests from the Spring front controller (the DispathcerServlet)**
- **After completing the requests it sends back the HTTP response**
- **Annotated with @RestController which is a combination of:**
  - **@Controller**
  - **@ResponseBody**
- **In a Spring Boot application it should render (serialize) JSON response;**

https://www.tutorialspoint.com/springmvc/springmvc_overview.htm

Routing

- The @RequestMapping annotation provides “routing” information based on the URI
- It tells Spring that any HTTP request with the path “/” should be mapped to the home method
- The HTTP method parameter has no default – so if we don’t specify a value, it’s going to map to any HTTP request.
- HTTP Methods variants:
  - @GetMapping
  - @PostMapping
  - @PutMapping
  - @DeleteMapping
  - @PatchMapping
- At the class level an @RequestMapping is more useful for expressing shared mappings.