Spring HTTP and REST API

Nikola Rakić
rakic.nikola@nsoft.com

POWERED BY:

nsoft sportradar

v1.0.0
2019-10-03

Introduction

Spring HTTP and REST API course will take you through the basic concepts that are behind all RESTful web services. You will learn how to build a RESTful web service using Spring and Spring Boot.

- Jump straight into the example project
- A brief HTTP overview
- A brief introduction to Web services
- An introduction to RESTful APIs
- Let's implement the example projects step by step

A quick demo

Jump into the main example project!

- *Todo list web service*
- Start the Spring Boot application
- Server running on *http://localhost:8081*
- Let's explore the API

Demo project RESTful API

- **GET /users** - Retrieve collection of users
- **POST /users** - Create a new user
- **GET /users/{id}** - Get a single user
- **PUT /users/{id}** - Update a user
- **DELETE /users/{id}** - Remove a user

Demo project RESTful API

- **GET /users/{userId}/todos** - Retrieve a collection of todos for a user
- **POST /users/{userId}/todos** - Create a new todo for a user
- **GET /users/{userId}/todos/{id}** - Retrieve a single todo for a user
- **PUT /users/{userId}/todos/{id}** - Update a single todo for a user
- **DELETE /users/{userId}/todos/{id}** - Remove a todo for a user

HTTP

HTTP overview

*"The **H**yper**t**ext **T**ransfer **P**rotocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems."*

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

HTTP overview

"HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the client and an application running on a computer hosting a website may be the server."

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

HTTP overview

"The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body."

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

HTTP key concepts recap

- Resource
- Server
- Client (User agent)
- Request
- Response
- Message body
- Headers
- Status codes

HTTP headers

Example headers:

- Content-Type: application/json
- Set-Cookie: <cookie-name>=<cookie-value>
- Location: http://spark.ba

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

HTTP request methods

- GET
- POST
- PUT
- DELETE
- Other

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

HTTP response status codes

- Informational responses (100–199),
- Successful responses (200–299),
- Redirects (300–399),
- Client errors (400–499),
- and Server errors (500–599).

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

CURL example

- Open your terminal and execute:
- `curl -i -X GET https://accounts.google.com/.well-known/openid-configuration`

https://en.wikipedia.org/wiki/CURL

Web services

Web service definition

- A client-server application built for the interoperable machine-to-machine communication over the web using a collection of standards for information exchange.

Web service key features

- Designed for M2M (A2A) interaction
- Interoperable (platform independent)
  - Use standardized request/response format
- Allows communication over network
  - Exposes an HTTP API

Types of web services

- SOAP web services
- RESTful web services

POWERED BY: nsoft sportradar

SPARK. SCHOOL

REST

REST overview

*"**Re**presentational **s**tate **t**ransfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services."*

https://en.wikipedia.org/wiki/Representational_state_transfer

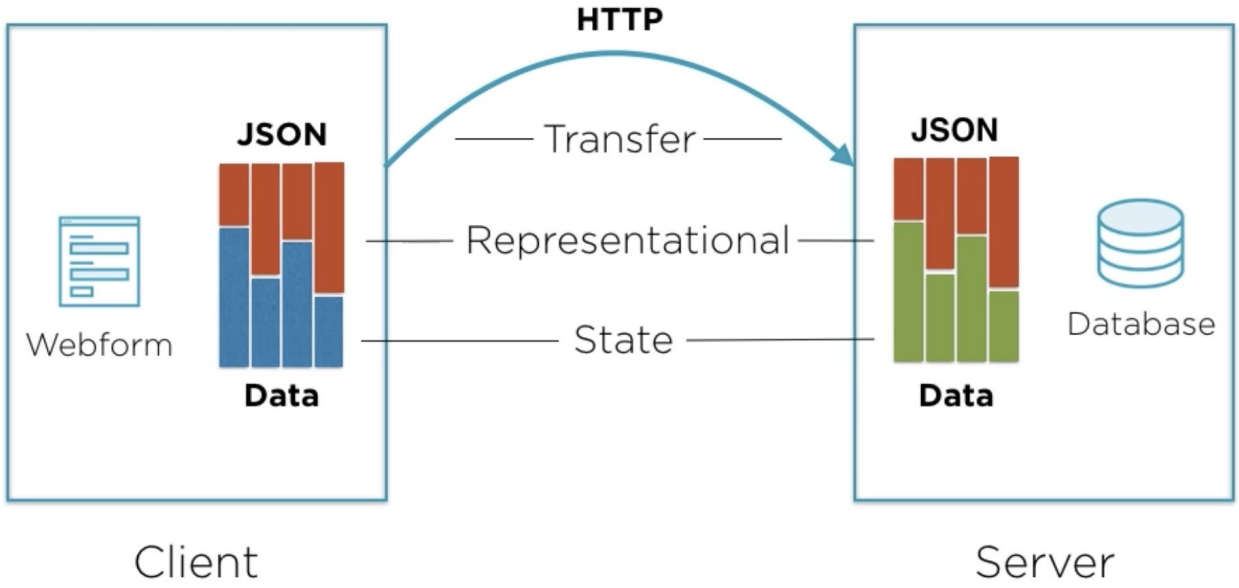
[...] defined in 2000 by Roy Fielding in his doctoral dissertation.

https://en.wikipedia.org/wiki/Roy_Fielding

REST - key points

- Uses HTTP for transport
  - Headers, Methods, Body, Statuses
- Resource as a main abstraction
  - URL
- Different representations possible
  - XML, JSON, etc.
- Web service definition/documentation
  - WADL - obsolete
  - Swagger - widely adopted

REST - key points



POWERED BY: nsoft sportradar

SPARK. SCHOOL

It’s Spring time

Using Spring "*magic ingredients*"

- *@RestController*
- *@RequestMapping*
- *@PathVariable*
- *@RequestParam*
- *@RequestBody*
- *@ResponseStatus*
- *RequestEntity class*
- *MediaType class*
- *HttpStatus class*
- *@Component*
- *@Autowired*

Hello world example

- What is your “java --version”?
- Let’s initialize our Spring Boot project: https://start.spring.io/.
  - Group id: ***ba.spark.bootcamp.rest***
  - Artifact id: ***helloworld***
  - ***Spring Web*** dependency
- Download, unarchive and open the project in your IDE.
- Let’s code our *Hello world* example!

Hello world example

Let's do the following:

- Server running on http://localhost:8082
- Create a new controller,
- Expose a single API route "*GET /helloworld*",
  - Output default "Hello World!" message in response

Hello world example

Let's do the following:

- Change the representation format to JSON
- Add a path variable {name}: "GET */helloworld/{name}*",
  - Output "Hello World, {name}!" message in response
- Add request parameter {lowercase}: "*?lowercase=true*"
  - Output "hello world, {name}!" message in response

Todo List example

RESTful API:

- GET /users
- POST /users
- GET /users/{id}
- PUT /users/{id}
- DELETE /users/{id}

- GET /users/{userId}/todos
- POST /users/{userId}/todos
- GET /users/{userId}/todos/{id}
- PUT /users/{userId}/todos/{id}
- DELETE /users/{userId}/todos/{id}

Source code

Example source code will be provided after each lecture:

- https://github.com/nsftx/java-bootcamp-2019/tree/beginner/module-6/