Weather App Course Notes

MYSQL DB Password

Password

User

Root or user

# What are the benefits of REST API’s

<https://restfulapi.net/statelessness/>

REST API’s are not a framework, but a architectural format to building web API’s.

They are stateless, which means that each request provides enough information to be processed by itself without data from a previous request. Treat every request as if it’s a new request. Pass in the authorization you need for each request. Server holds onto no logic and neither does the client (I don’t think)

# What are URI’s

<https://www.udemy.com/course/spring-boot-rest-apis-ultimate/learn/lecture/32542268#overview>

What is the difference between client error 400-499 vs internal server error 500-599

<https://medium.com/@rathin91/difference-between-client-side-and-server-side-http-status-error-codes-7f8a35662f0d#:~:text=In%20summary%2C%20the%204xx%20class,to%20internal%20issues%20or%20unavailability>.

“In summary, the 4xx class of HTTP status codes indicates client-side errors, usually caused by problems with the client’s request or authentication, while the 5xx class indicates server-side errors, which occur when the server is unable to fulfill the client’s request due to internal issues or unavailability.”

# API Versioning

<https://semver.org/>

you should always version your API’s, following the major, minor and patch semantics. Major is when you do a non backward compatible changes. Minor is for normal feature changes that are backwards compatible, patch is when you do a bug fix that is backwards compatible

<https://www.postman.com/api-platform/api-versioning/>

<https://www.google.com/search?q=why+do+we+need+to+version+APIs&oq=why+do+we+need+to+version+APIs&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQIRigATIHCAIQIRifBTIHCAMQIRifBTIHCAQQIRifBTIHCAUQIRifBTIHCAYQIRifBTIHCAcQIRifBTIHCAgQIRifBdIBCDMyOTdqMGo3qAIAsAIA&sourceid=chrome&ie=UTF-8#fpstate=ive&vld=cid:778a4632,vid:vsb4ZkUytrU,st:0>

# API Caching

# API Rate Limiting

<https://learn.microsoft.com/en-us/partner-center/developer/api-throttling-guidance>

Rate limiting lets you control how many times a user can query your API, this ensures fair usages for all users and also helps for security reasons such as malicious bot attacks or DDOS attempts.

API’s are finite resources, rate limiting ensures that they are used fairly for everyone and prevent throttling of your application This is done to prevent overload of your servers resources.

# API Service System Architecture

A diagram of a computer network

Description automatically generated

# OpenAPI Specification

<https://github.com/OAI/OpenAPI-Specification>

A screenshot of a computer program

Description automatically generated

# Steps to implement a new API

A screenshot of a computer program

Description automatically generated

# Spring Boot Controller Testing

A screenshot of a computer program

Description automatically generated

# Future Spring Boot related questions

* What happens when you build your project?
* How do you update your dependencies in the pom.xml when you add new ones?
* What is hibernate doing under the hood?
* What is spring tool suite doing that is different to eclipse as an IDE?
* What is the ResponseEntity object doing?
* What are builders in spring boot?
* What are records in spring boot?
* What are some of the feature of Java 17?
* When you call CRUDREPOSITORY, WHAT ARE THE TWO GENERICS THAT NEED TO BE PASSED TO IT? CAN’T REMEMBER WHAT THEY NEED TO BE
* The @service and @component annotations provide beans, does the @controller annotation give the class its own bean?
* When you serialise a Java object to JSON, is this a JSON string It gets serialised to?
* What is the difference between @WebMvcTest vs @SpringBootTest
* Should you only use property level object injection during tests? I know that constructor level injection should only happen for normal class files.
* What response do you return due to an invalid request from the user?
  + <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#client_error_responses> (400 is bad request
* What does nullable mean in database entry column?
* Do you need getters + setters for repository layer retrieval? I remember scott wheeler saying something about getters and setters are used for serialisation
* How do you write tests for the repository layer? Should you use a h2 database for testing this?
* Should you write tests for your service layer?

## Course TODO:

* Why is the @NotBlank annotation not working on the ‘Unit testing for REST API’ section?
* How to fix mvn commands, I have it installed but none of them work at the moment. My workaround is to right click project and select maven -> update project, select them all and choose force update and snapshot
* What is jsonpath expression?
* Why is hashcode and equals getting rid of the 201 error scenario on the test?

## Running Controller Tests

A screen shot of a computer program

Description automatically generated

To run the controller tests, you need to right click the method name and choose Run as -> Junit Test