SPRING BOOT REST CRUD APIS

What are rest services?

* **RE**presentational **S**tate **T**ransfer = REST
* We can make REST API calls over HTTP.
* Lightweight approach for communicating between applications.
* Rest is language-independent so we can use any language.
* Any programming language on the client and server side is acceptable.
* Can use any data format but JSON is more popular than XML.
  + JSON JavaScript Object Notation

What is JSON?

* JSON is the JavaScript Object Notation
* A lightweight data format for storing and exchanging data.
* It’s plain text.
* Can be used with any programming language not just JavaScript.
* Curly braces define objects in JSON.
* Object members are name/value pairs delimited by colons.
* The name is always in double-quotes.

Spring Boot REST HTTP Basics

* The most common use case for REST is over HTTP
* Leverage HTTP methods for CRUD operations
* HTTP Methods and their CRUD operations:
  + POST request – Create a new entity.
  + GET request – Read a list of entities or a single entity.
  + PUT request Update an existing entity.
  + DELETE request – Delete an existing entity.
* HTTP Request Message
  + 3 main parts:
    - Request Line – the HTTP command (GET, POST, or DELETE).
    - Header variables – request metadata which is additional data about the request.
    - Message body – contents of the message (payload).
* HTTP Response message
  + 3 main parts:
    - Request Line – Server protocol and status code (200, 404, or 500)
    - Header variables – response metadata. What is the content type of the metadata? is it XML or JSON? What’s the size or length of the data?
    - Message body – contents of the message (payload). Will come back as XML or JSON depending on the server’s configs.
  + Status code ranges and their description:
    - 100-199 = Informational
    - 200-299 = Successful
    - 300-399 = Redirection
    - 400-499 = Client error
    - 500-599 = Server error

Postman

* Postman
  + Jsonplaceholder.typicode.com – Free fake API for testing and prototyping REST services

Spring Boot REST Controller

* Web browser vs Postman
  + For simple REST testing for GET requests, they’re both similar.
  + The Postman application has much better support for advanced REST testing such as POST, PUT, etc.
  + Postman is better for POSTing JSON data and setting the content type.
  + Passing HTTP request headers, authentication, etc.…

Java JSON data binding

* Data binding is the process of converting JSON data to JAVA POJO or vice versa.
* Java POJO is a plain old Java object or any old Java class.
* Data binding could also be known as mapping, sterilization, desterilization, marshaling, or un-marshaling.
* Jackson is Spring Boots data binding project behind the scenes, but Jackson is a separate project from Jackson.
* Jackson will call the appropriate getter or setter methods whether you’re converting from JSON to Java POJO or Java POJO to JSON.
* To convert JSON to Java POJO Jackson calls setter methods on Java POJO
* To convert Java POJO to JSON Jackson calls getter methods on Java POJO
* When you’re building Spring REST applications Spring will automatically handle the Jackson integration.
* Any data being passed to the REST controller is converted to a POJO.
* Any Java object being returned from the REST controller is converted to JSON.

Spring Boot REST Path Variables

* We can call our variable by using curly braces at the end of our URL.

Spring Boot REST Exception Handling

* We need to implement REST exception handling to better manage the passing of requests to the server with bad data.
* When we run into an error, we want to handle it by returning the error as JSON.
* It’s desirable that this error has a status, message, and timestamp.

Spring Boot REST Global Exception Handling

* Best practice
* Exception handler code is only for the specific REST controller and can’t be used by other controllers. Large projects will have multiple controllers.
* What we need to combat standard exception handler limitations are global exception handlers which promote reuse and centralize your exception handling and minimize the amount of code that you need to duplicate across multiple controllers on very large projects.
* Programmers can use the @ControllerAdvice which is like an interceptor or a filter that we can use to pre-process requests to controllers, and post-process responses to handle exceptions, and it’s perfect for global exception handling.
* The major key is that now we would have global exception handlers, so the code is no longer stuck in the service itself. We can actually make use of it at the controller advice level, and we could have multiple services that could all throw exceptions.

Spring Boot REST API Design - Best Practices

* Things to consider when designing an API:
  + Who will use the API?
  + How will they use the API?
  + What are the requirements?
* API Scenario: I must create a REST API for the employee Directory…
  + Design process:
    - Review API requirements.
      * REST clients should have CRUD capabilities and be able to:
        + Get a list of employees.
        + Get a single employee by id.
        + Add a new employee.
        + Update an employee.
        + Delete an employee.
      * Identify the main resource/entity.
        + To identify the main resource/entity, look for the most prominent “noun”.
        + For our project, the most prominent noun is “employee”.
        + The convention is to use the plural form of resource/entity: employees.
      * Use HTTP methods to assign the action on a given resource.
        + POST request – Create a new entity.
        + GET request – Read a list of entities or a single entity.
        + PUT request Update an existing entity.
        + DELETE request – Delete an existing entity.
        + Real-time HTTP Methods:

POST request – /api/employees

GET request – /api/employees

GET request – /api/employees/ {employeeId}

PUT request – /api/employees

DELETE request – /api/employees/ {employeeId}

* + - * + DO NOTs w/ Real-time HTTP Methods:

Don’t include actions in the endpoint for example if the endpoints CRUD operation is to delete or add do not do this: /api/deleteEmployee or /api/addEployee/ {employeeId}