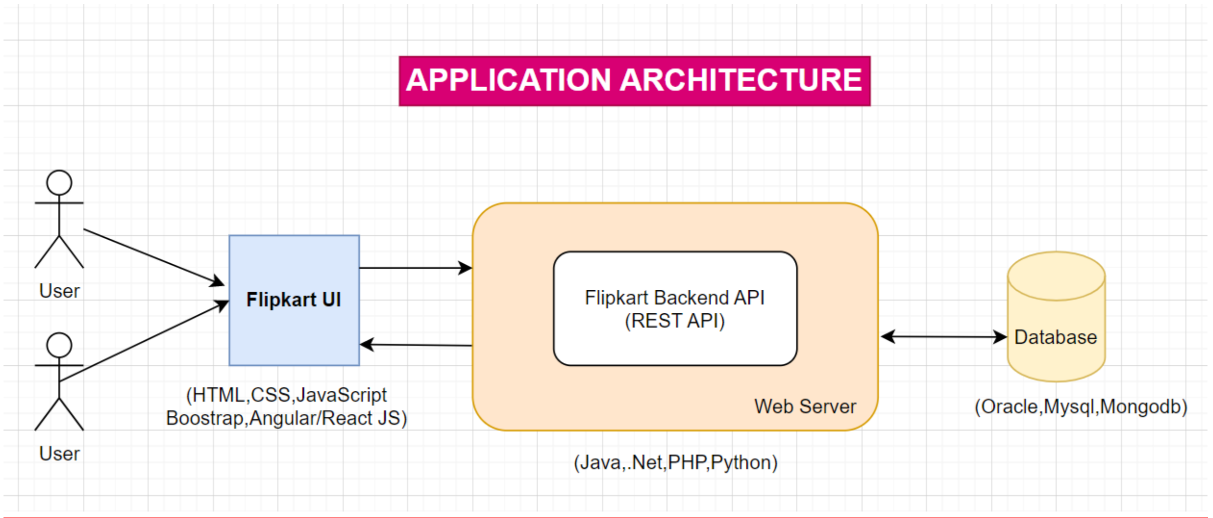
www.youtube.com/c/javaexpress
+91 7801007910
Java Express

Restfull Webservices



Web service:

- A web service is a technology that enables communication and data exchange between different software applications over the internet
- A web service is technology which is used to develop distributed applications with interoperability.

Benefits of Web Services:

- Irrespective of platform irrespective of programming language if two applications are communicating with each other then they are called as Interoperable applications.
  - Java ←→Angular
  - Dot Net – Python
- **Reusability :**
  - Web services can be reused by multiple applications, minimizing development effort.

Examples of Web Services:

- Weather services providing weather data to various apps.
- Payment gateways processing online transactions.
- Social media APIs allowing integration with third-party applications.
- Cloud storage services offering programmatic access to stored data.

Key Components:

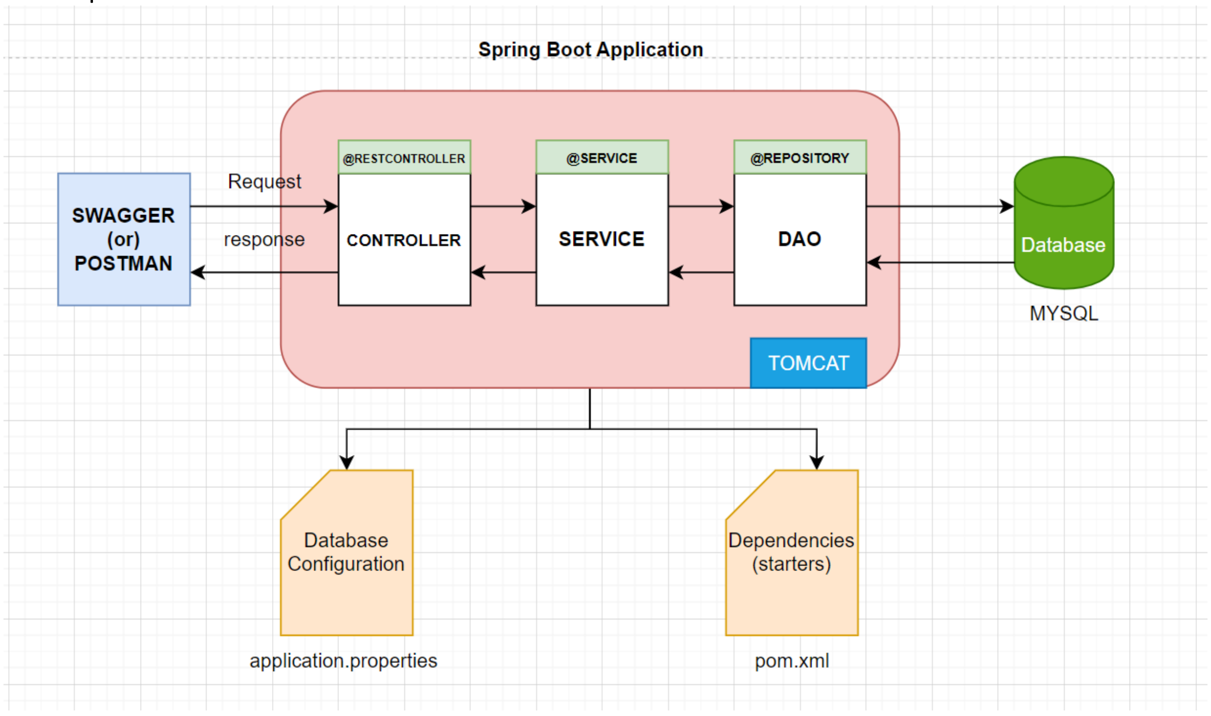
- **Communication Protocols**
  - Web services typically use standard communication protocols such as HTTP (Hypertext Transfer Protocol) or HTTPS (HTTP Secure) to transmit data over

the web. These protocols ensure that data can be exchanged reliably and securely.

- **Data Format:**
    - Web services use **standardized** data formats for representing information. Common formats include XML (Extensible Markup Language) and JSON (JavaScript Object Notation). These formats ensure that data can be understood by different systems.
- **Service Description:**
    - Web services are often described using a standard language like WSDL (Web Services Description Language) or Open API, which provides a formal description of how the service can be accessed, what operations it performs, and what data it expects and returns.
- **Service Endpoints:**
    - A web service is hosted on a server and has specific endpoints (URLs) that clients can use to access its functionality. These endpoints are defined in the service description.

<u>Rest Architecture:</u>

- In RESTFUL Services architecture 2 actors will be involved
    - Resource
    - Client
- Resource will provide business service to other applications.
- Client will access business services from other applications.
- Resource Development team will provide documentation to client-side team to understand Resource details.
        - Open API (Trending)
- Spring web MVC jars are sufficient to develop Restful services (additional jars not required)



Http Protocol

- In Restful services, two actors will be available
    1) Resource
    2) Client
- To Establish communication between Client and Resource we will use HTTP protocol
- For HTTP Protocol we have 2 versions they are
    1) HTTP 1.0 (status codes support is not available)
    2) HTTP 1.1
- In Http protocol we have several methods like below
    1) GET (read only request)
    2) POST
    3) PUT
    4) DELETE

- **GET** METHOD is used to retrieve information from the Server using given URI. (The data will be appended to URL in GET Request (path params & query params)
- **POST** method is used to send information to server in request body (File upload, HTML form) - it creates new record at server.
- **PUT** Method is used to replace all the current representation of record/resource.(current data will be updated with new data)
- **DELETE** method is used to remove current representation of the target record/resource from the server which is given by URI.

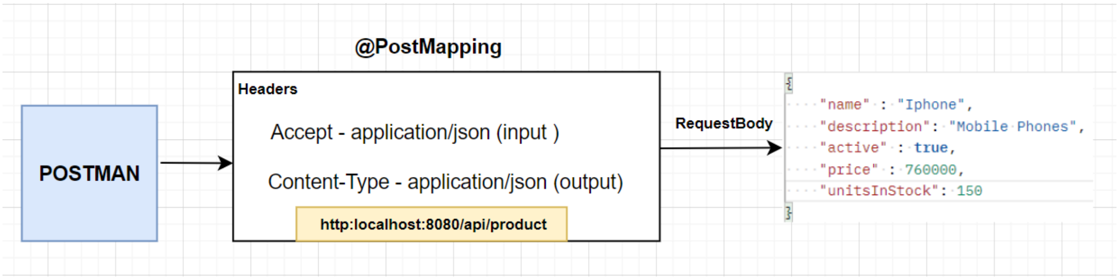
Spring MVC & Rest Annotations

@RestController:

- It is a spring annotation that marks a class as a controller that handles Restful web service requests. It combines the functionality of two other annotations
- @Controller: Indicates that a class is a controller, responsible for handling HTTP requests and generating responses.
- @ResponseBody: Applied to methods within a controller, indicating that the return value should be directly written to the HTTP response body, rather than rendering a view.

@PostMapping:

- The @PostMapping annotation is used to map the HTTP POST request on the specific handler method.
- It is an alternative of @RequestMapping (method = RequestMethod.POST).
- POST method is used to send information to server in request body (File upload, HTML form) - it creates new record at server.
- Http Post Request will have request body and client needs to send along with it
- Post Request means creating a new record at server. If Post Request operation is successful it should represent with HTTP 201 status code.


@PostMapping
POSTMAN
Headers
Accept - application/json (input)
Content-Type - application/json (output)
http:localhost:8080/api/product
RequestBody

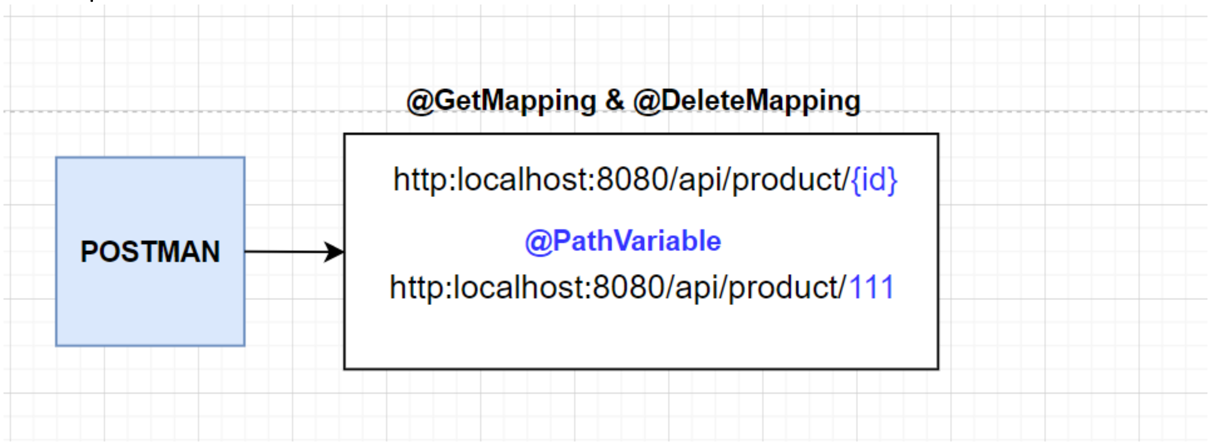
```
{
    "name" : "Iphone",
    "description": "Mobile Phones",
    "active" : true,
    "price" : 760000,
    "unitsInStock": 150
}
```

@RequestBody:

- The @RequestBody annotations are used to bind HTTP request with an object in a method parameter.
- Internally it uses **HTTP Message Converters** to convert the request's body.
- When a method parameter is annotated with @RequestBody annotation, the Spring framework wraps the incoming HTTP request body to that parameter.

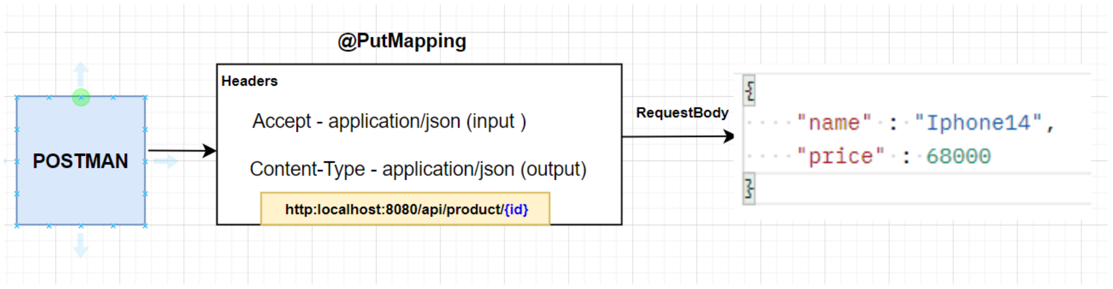
@GetMapping:

- The @GetMapping annotation is used to map the HTTP GET request on the specific handler method.
- It is an alternative of @RequestMapping (method = RequestMethod.GET)
- GET Method is used to retrieve information from the Server using given URI.


@GetMapping & @DeleteMapping
POSTMAN
http:localhost:8080/api/product/{id}
@PathVariable
http:localhost:8080/api/product/111

@PutMapping:

- The @PutMapping annotation is used to map the HTTP PUT requests on the specific handler method.
- It is useful for creating a web service endpoint that creates or updates. It is an alternative of @RequestMapping (method = RequestMethod.PUT).
- If our rest controller method is responsible for updating existing record data then we will bind that method to PUT Request method.


@PutMapping
POSTMAN
Headers
Accept - application/json (input)
Content-Type - application/json (output)
http:localhost:8080/api/product/{id}
RequestBody

```
{
    "name" : "Iphone14",
    "price" : 68000
}
```

@PathVariable:

- The @PathVariable annotation is used to extract the values from the URI.
- It is the most suitable annotation for building the Restful web service, where the URL contains a path variable.
- Path Parameters are used to send data to server in URL
- Path Parameters will not contain any key
    - **http://localhost:9090/product/{id}**
- Path Parameters can present anywhere in the URL
- When Rest Resource method is expecting Path Parameter then it should be represented in URL pattern

@DeleteMapping:

- If our rest controller method is responsible for delete a record at server then we will bind that method to DELETE Request method.
- Only for GET, request body is not available. For POST, PUT and Delete we have request body.

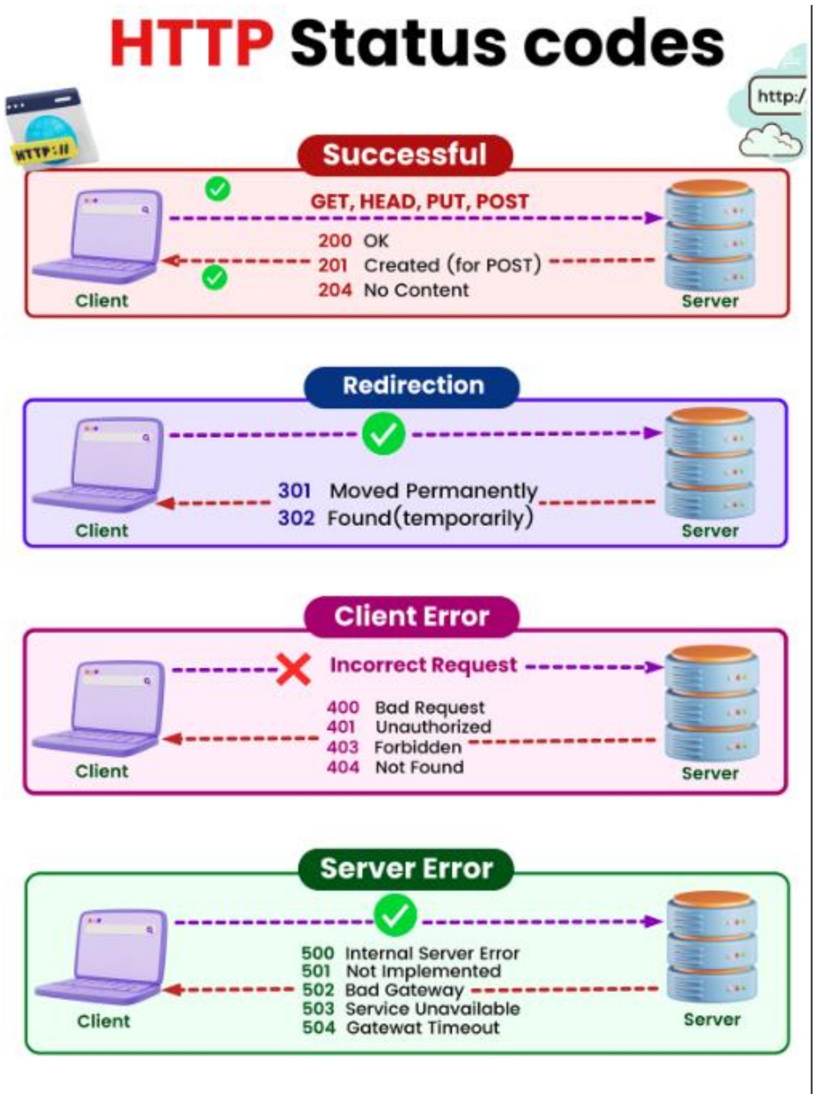
@RequestParam:

- The @RequestParam annotation aka query parameter is used to extract the query parameters from the URL.
- It is most suitable for developing web applications. It can define the default values if the query parameter is not present in the URL.
- Query Parameters are used to send the data to server in the URL
- Query Parameters will represent data in the form of Key-Value Pair
- Query Parameters will start with '?'
- Query Parameters will be separated by '&' symbol
- Query Parameters should present only at end of the URL
- In Spring MVC to read query parameters we will use
    - @RequestParam("productId") Integer productId
    - **http://localhost:8080/product?**productId**=45**

Http Status Codes & Messages:

- When client sends http request to server, it will process that request and it will send response back to client with status code, status message, headers & response body.
- Status code status message protocol version
- Http status codes are divided into 5 categories
    - 1XX (100-199): Information details
    - 2XX (200-299): Success
    - 3XX (300-399): Redirectional
    - 4XX (400- 499): Client Error

        400 - Bad Request
        404 - Wrong URL
        405 - Method Not Allowed
        406 - Not Acceptable
        415 - Unsupported Media Type
        5XX (500- 599): Server Error



Open API Documentation: (Swagger)

- OpenAPI Swagger documentation refers to the practice of using Swagger tools to create, visualize, and interact with API documentation that adheres to the OpenAPI Specification (OAS). It provides a powerful and user-friendly way to describe, manage, and consume Restful APIs.
- **OpenAPI Specification (OAS):** Standardized language for defining Restful APIs.
  - Describes API endpoints, operations, parameters, responses, authentication methods, and more.
  - Uses YAML or JSON format for portability and readability.
- **Swagger:** A set of open-source tools for working with the OpenAPI Specification.
  - Swagger Editor: Online/offline tool for creating and editing OAS documents.
  - Swagger UI: Interactive, browser-based interface for exploring and testing APIs.
- Swagger documentation Contains below details
  - Resource method Types
  - Resource method URLs
  - Resource Method Consumes types

  - Resource Method Produces types
  - Resource Method Input Data Structure
  - Resource method Output Data Structure

Maven Dependency

```
    <dependency>
      <groupId>org.springdoc</groupId>
      <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
      <version>2.3.0</version>
    </dependency>
```

Properties

```
springdoc.swagger-ui.path=/swagger-ui.html
springdoc.packagesToScan=com.package1, com.package2
springdoc.pathsToMatch=/v1, /api/balance/**
springdoc.swagger-ui.path=/swagger-ui.html
```

How to access Swagger URL:

- http://localhost:8080/swagger-ui.html

Spring Profiles:

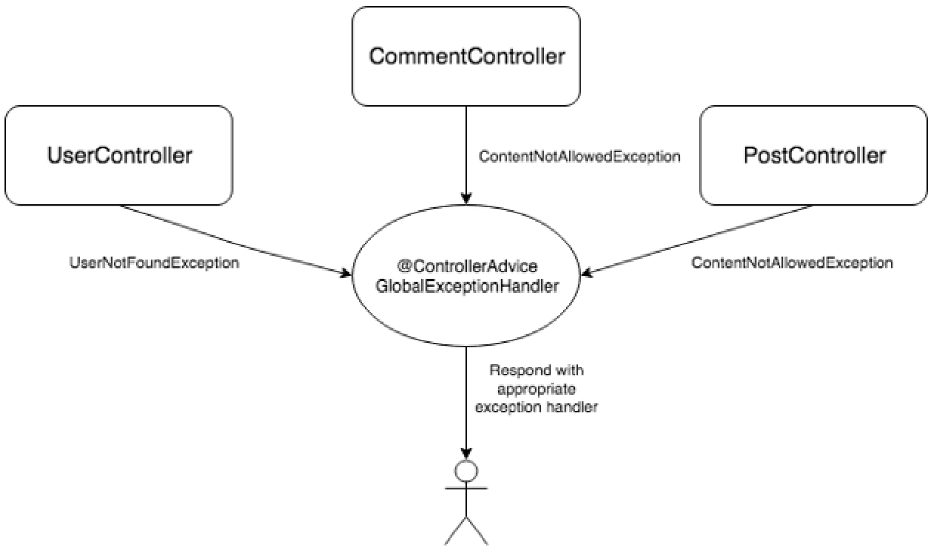
- Enterprise Spring applications typically need to run in multiple environments.
  - For example, development, testing, staging, and production
- . Each such environment has its own set of settings.
  - For example, you might be using an embedded H2 database for development but an enterprise-grade Oracle or MySQL database for production.
- Spring applications typically have a large set of such configurations that vary across environments.
  - In such scenarios, you can use Spring profiles.

Spring Profiles Creation:

- Spring Boot by default comes with a property file, named application.properties
- To segregate the configuration based on the environments, we will create multiple property files. One for each environment we are targeting.
- We will create three property files for the dev, test, and prod environments. Note the naming convention.
  - application-dev.properties
  - application-test. properties
  - application-prop. properties
- The application.properties file will be the master of all properties. Here we will specify which profile is active by using the property
  - spring. profiles. active = dev

Exception Handling in SpringBoot:

- @RestControllerAdvice used for global error handling in the Spring MVC application.
- It allows you to handle exceptions across the whole application, not just to an individual controller and also has full control over the body of the response and the status code.
- GlobalExceptionHandler was annotated with @RestControllerAdvice, thus it is going to intercept exceptions from controllers across the application.


CommentController
UserController
ContentNotAllowedException
PostController
UserNotFoundException
@ControllerAdvice GlobalExceptionHandler
ContentNotAllowedException
Respond with appropriate exception handler

- The @ExceptionHandler is an annotation used to handle the specific exceptions and sending the custom responses to the client.

Development Steps

- Create your own exception classes:
- One Controller advice class per Application
- Write a @ ExceptionHandler method
- Handle the Status Code based on our needs

H2 Database:

- H2 is an embedded database provided by Spring Boot
- H2 is a relational database management system written in java.
- It is in-memory database.
- H2 provides web console to maintain the database (or) Browser based console application.
- H2 database is available when you start your spring boot application and it will drop when you stop the spring boot application.
- It is generally used in unit testing.
- It stores data in memory, not persist the data on disk.

Rest Client:

- Any application/Tool/Human which is cable of Sending HTTP Request can act as REST Client.
    - Python project can act as REST Client
    - Net Project can act as REST client
    - Java Project Can act as REST Client
    - Angular & React Projects also can act as REST Clients
    - POSTMAN & Swagger-UI tools are acting as REST Clients.

How to Develop REST Client using JAVA

- Rest Client can be developed in 2 ways using Spring MVC
    1) Rest Template
    2) Web Client
- Rest Template supports Synchronous communication
    - After Making a request it will block the thread until we receive the response
- Web Client supports both synchronous and asynchronous Communication.
    - Asynchronous communication means after making a request it won't wait for the response it will continue its execution further.
    - Mono Same as RestTemplate – Single Request -> Single Response
    - Flux – Single Request -> Multiple Responses

Difference between SOAP vs REST

<u>SOAP (Simple Object Access Protocol):</u>

- Protocol-based: Defines a strict protocol for describing messages, data types, and operations.
- XML-based: Uses XML for data exchange, ensuring platform and language independence.
- Stateful: Maintains context between requests and responses for complex transactions.
- Security features built-in: Provides WS-Security standards for authentication and authorization.
- Mature and widely supported: Extensive tooling and libraries available.

<u>REST (Representational State Transfer):</u>

- Architectural style: Defines principles for designing web services based on HTTP methods and resources.
- Flexible data formats: Supports various formats like JSON, XML, plain text, etc.
- Stateless: Each request is independent and self-contained.
- Simple and lightweight: Easier to learn and implement compared to SOAP.
- Scalable and efficient: Less overhead due to stateless nature.
- Restful APIs are Simple: Used by the majority of modern web services and APIs.

www.youtube.com/c/javaexpress
+91 7801007910
Java Express