SPRING BOOT
&
RESTFUL WEB SERVICES

Aadil Anthrollil Abu- 245268

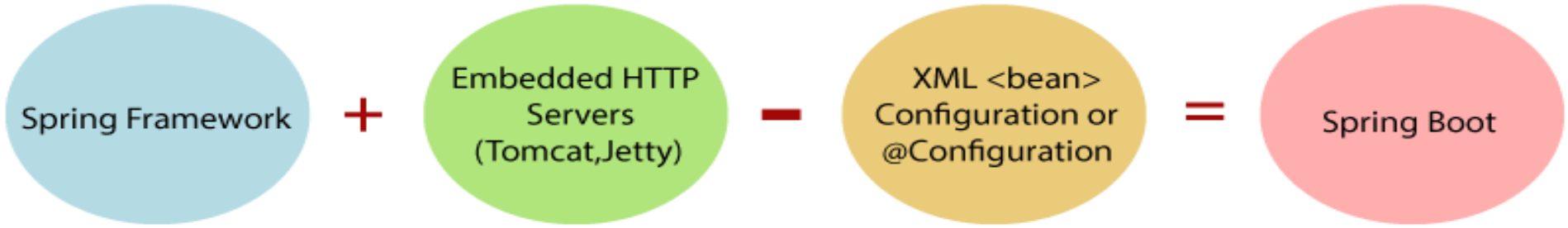
Christo Shaji-245052

Meera Javad-245217

Swetha S-245154

Introduction

- Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications.
- It is a Spring module that provides the **RAD (*Rapid Application Development*)** feature to the Spring Framework. It is used to create a stand-alone Spring-based application that you can just run because it needs minimal Spring configuration.



Why should we use Spring Boot Framework?

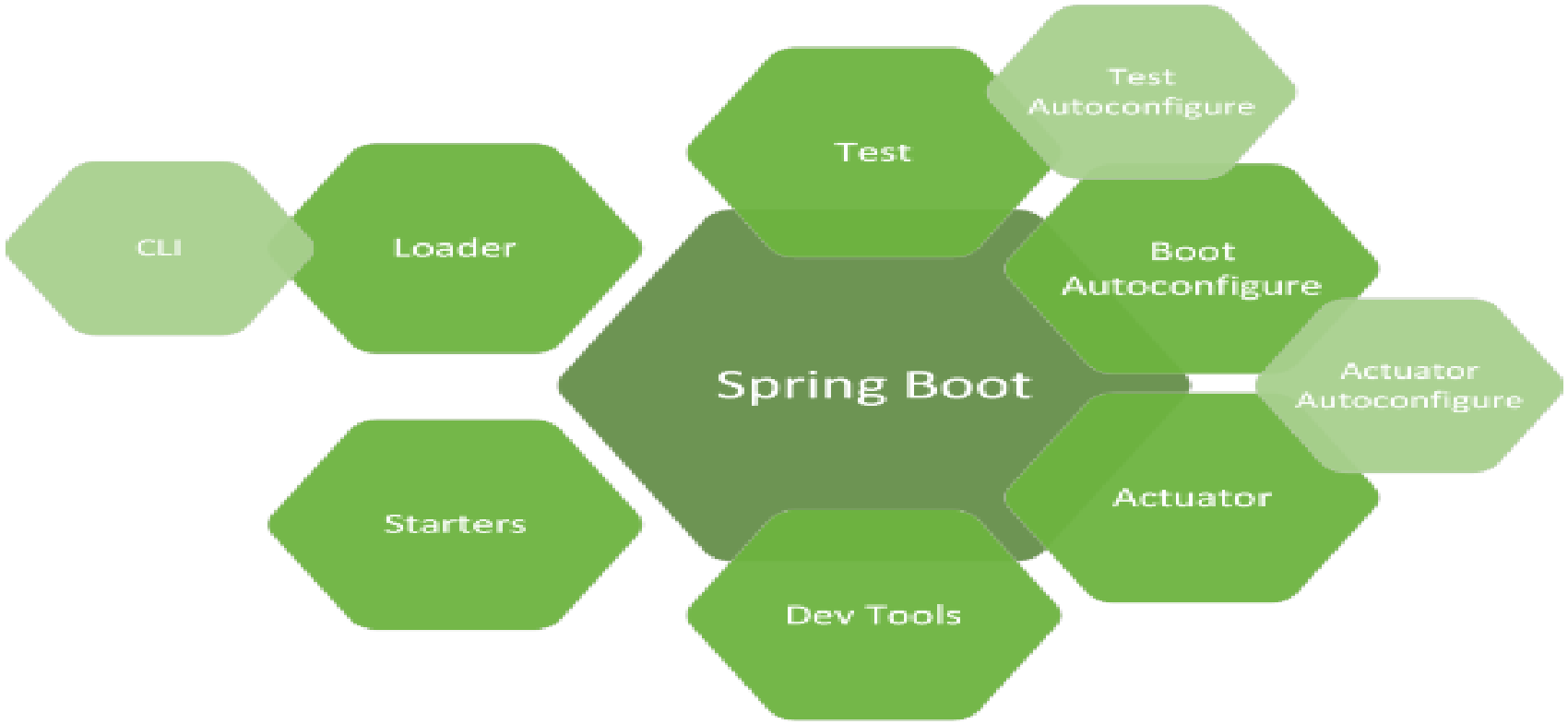
- The dependency injection approach is used in Spring Boot.
- It contains powerful database transaction management capabilities.
- It simplifies integration with other Java frameworks like JPA/Hibernate ORM, Struts, etc.
- It reduces the cost and development time of the application.

Advantages of Spring Boot

- It creates **stand-alone** Spring applications that can be started using Java **-jar**.
- It tests web applications easily with the help of different **Embedded** HTTP servers such as **Tomcat, Jetty,** etc. We don't need to deploy WAR files.
- It provides opinionated '**starter**' POMs to simplify our Maven configuration.
- It provides **production-ready** features such as **metrics, health checks,** and **externalized configuration**.
- There is no requirement for **XML** configuration.
- It offers a **CLI** tool for developing and testing the Spring Boot application.
- It offers the number of **plug-ins**.
- It also minimizes writing multiple **boilerplate codes** (the code that has to be included in many places with little or no alteration), XML configuration, and annotations.
- It **increases productivity** and reduces development time.

Spring Boot Modules

Spring boot provide various modules that improves overall efficiency of the application and helps in fast development of the project.



Limitation Of Spring Boot

Spring Boot can use dependencies that are not going to be used in the application. These dependencies increase the size of the application.

Goals of Spring Boot

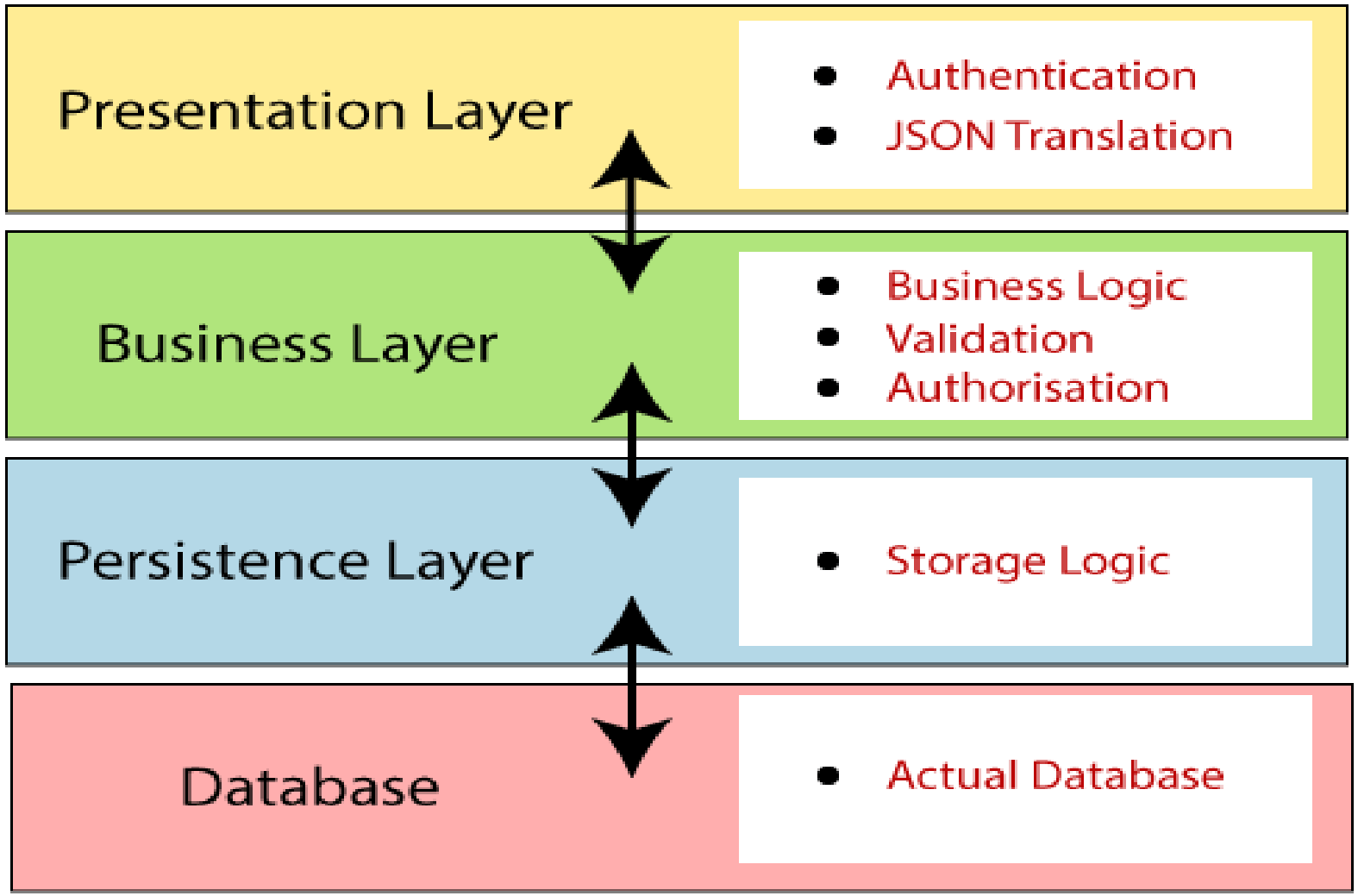
The main goal of Spring Boot is to reduce **development, unit test,** and **integration test** time.

- Provides Opinionated Development approach
- Avoids defining more Annotation Configuration
- Avoids writing lots of import statements
- Avoids XML Configuration.

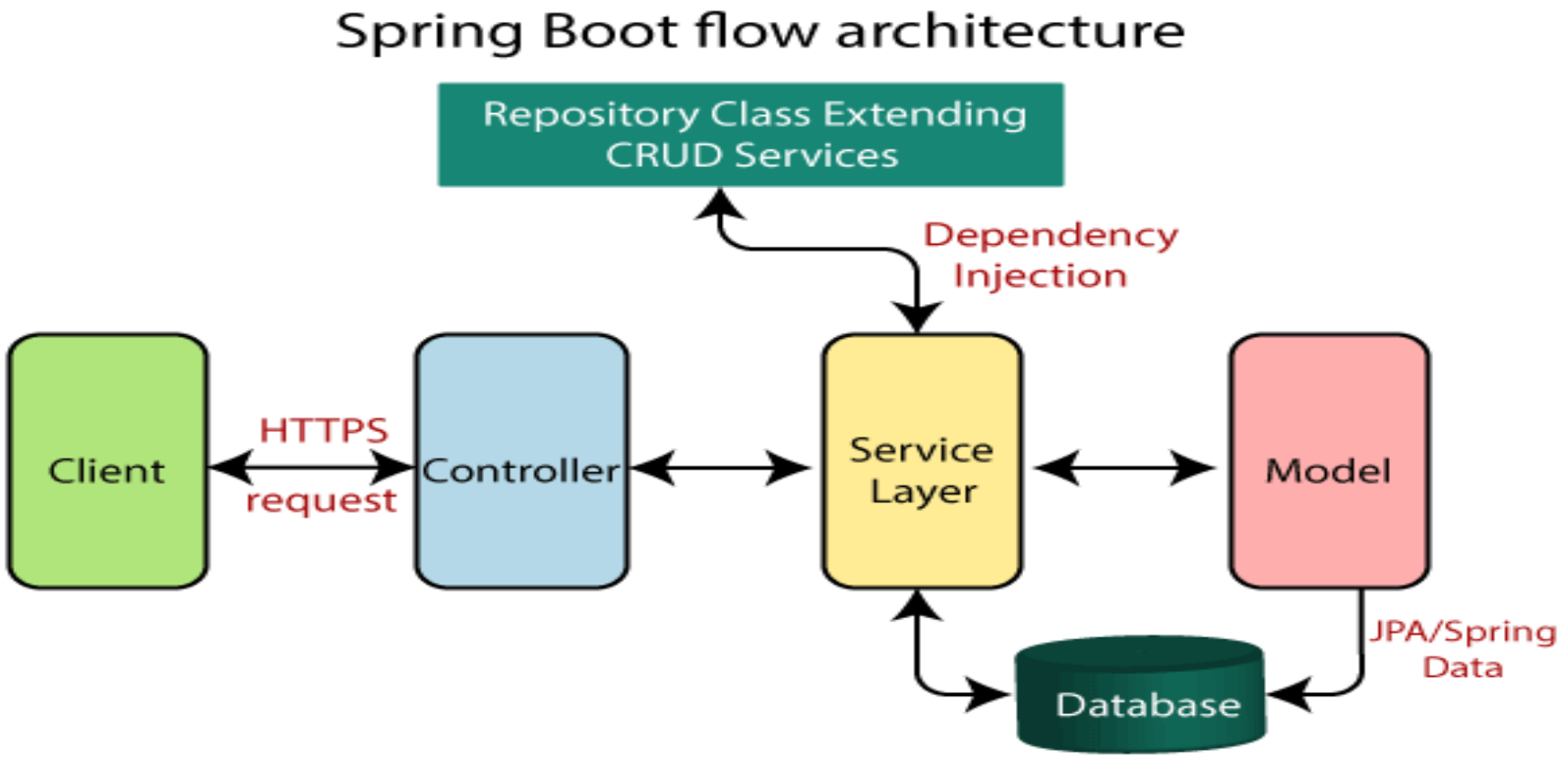
Prerequisite of Spring Boot

- Java 1.8
- Maven 3.0+
- Spring Framework 5.0.0.BUILD-SNAPSHOT
- An IDE (Spring Tool Suite) is recommended

Spring Boot Architecture



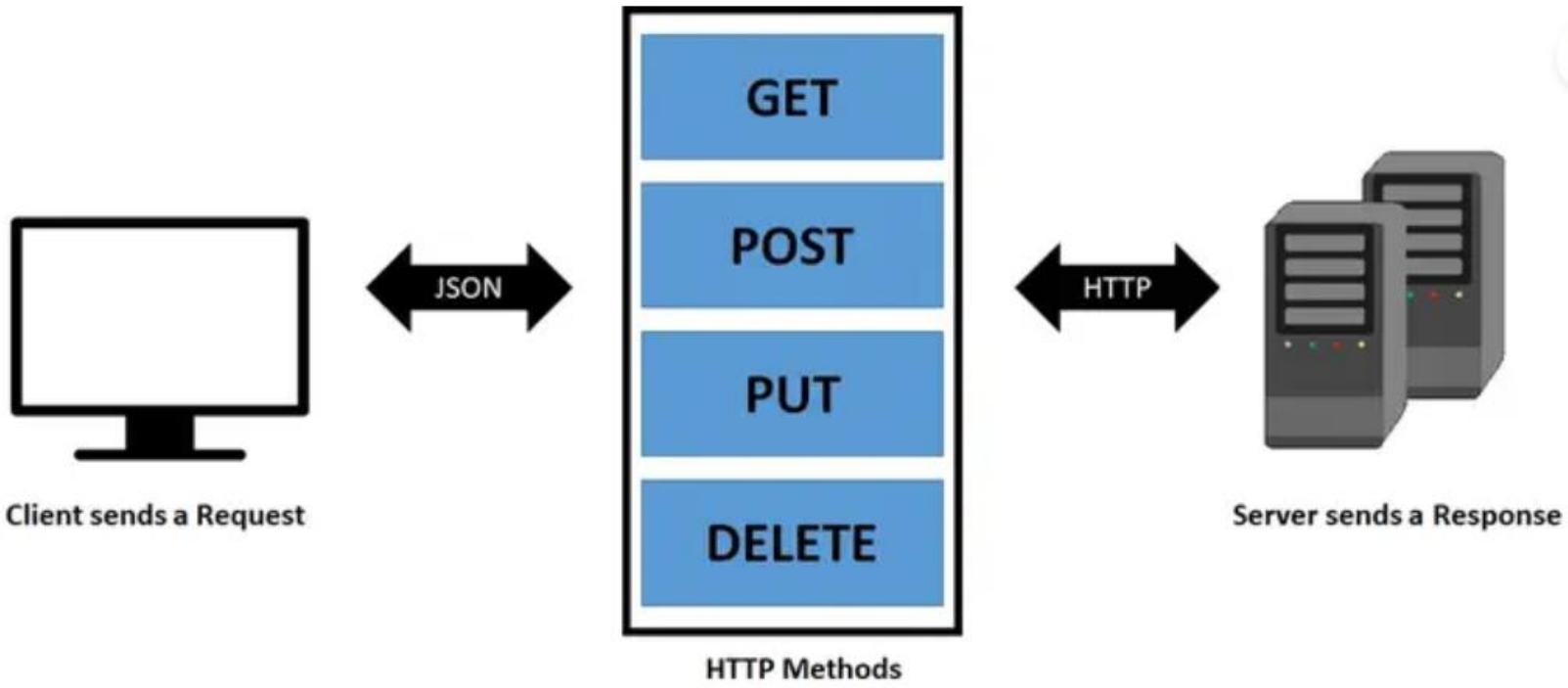
Spring Boot Flow Architecture



RESTFUL WEB SERVICES

Introduction

- RESTful web services are a type of web service that is based on the REST (Representational State Transfer) architecture.
- REST is an architectural style that defines a set of constraints to be used when creating web services.
- The REST architecture emphasizes the use of HTTP verbs (GET, POST, PUT, DELETE, etc.) to perform operations on resources, which are identified by unique URLs.



Attributes of Restful Web Services

1.Client-Server: It is a very important aspect of REST APIs. A REST API follows client-server architecture and these both should be separate. It means both the server and client can not be same server. In case it is same, you will receive CORS error.

2.Stateless: In REST, all calls are treated as a new call and any previous call state will not give any advantage to the new call. Hence during each call, it is required to maintain all the necessary authentication and other information.

3.Cache: A REST API encourages the browser and server caching process to enhance its processing speed.

4.Uniform Interface: The interface between the Client and Server remains uniform, hence any changes in either side will not affect the API functionality. This help in development of Client and Server system independently.

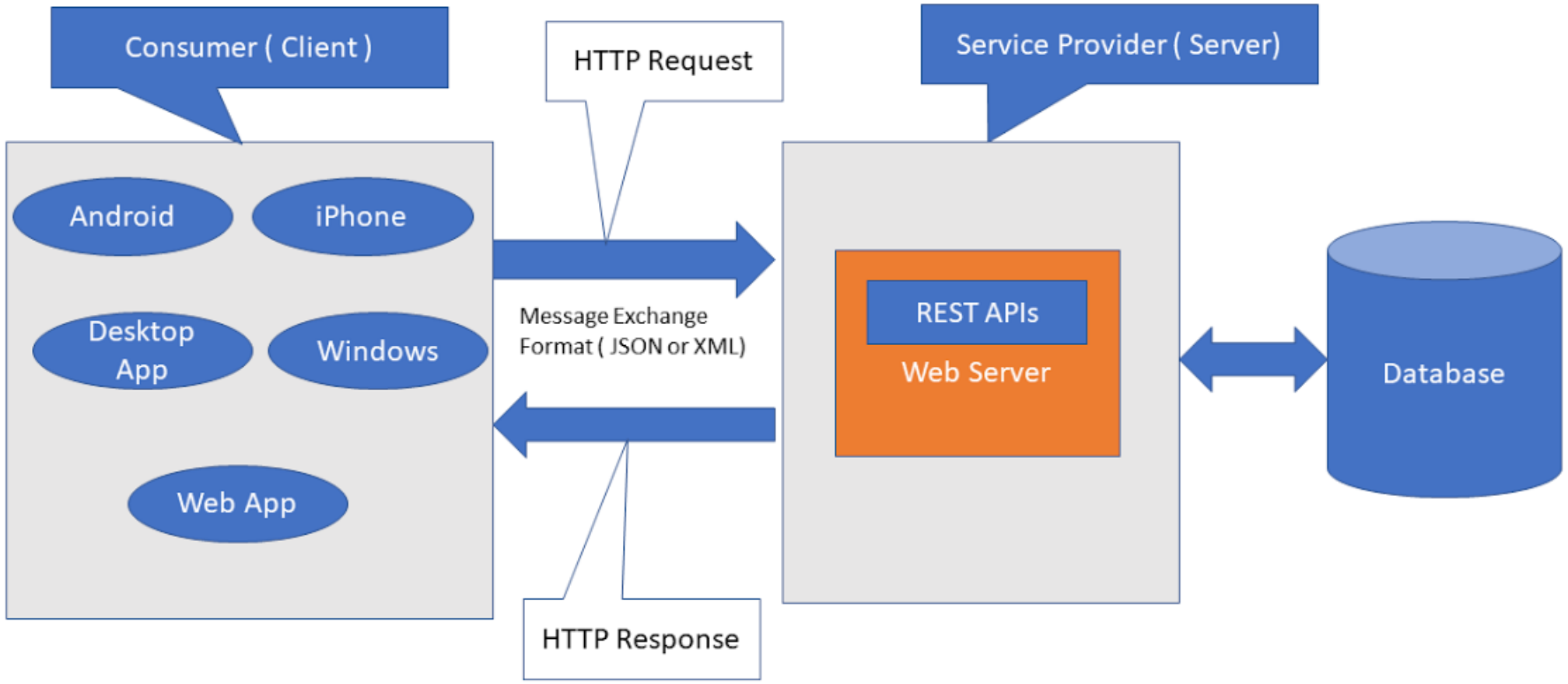
5.Layered System: REST allows usage of layered structure in server side i.e. you can have data on different server, authentication on different server while the API on different server. The client will never come to know that it is getting the data from which server.

6.**Code on Demand:** It is an optional feature of REST API where server can even send executable code to the client that can run directly during run time.

Methods in Restful Web Services

1. **GET:** This method is used to get a list of data from server.
2. **POST:** This method is used to post/create a new record in server.
3. **PUT:** This method is used to update an existing record of server.
4. **PUT**: This method is used to update an existing record of server.

REST – Architecture



Building RESTful Web Services

For building a RESTful Web Services, we need to add the Spring Boot Starter Web dependency into the build configuration file.

For Maven, add the following dependency in pom.xml file:

```
<dependency>
<groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

For Gradle, add the following dependency in build.gradle file:

```
compile('org.springframework.boot:spring-boot-starter-web')
```

Spring Boot Auto Configuration

•Spring Boot automatically configures a spring application based on dependencies present or not present in the classpath as a jar, beans, properties, etc.

•It makes development easier and faster as there is no need to define certain beans that are included in the auto-configuration classes.

- Auto-configuration can be enabled by adding:

  @SpringBootApplication or **@EnableAutoConfiguration** annotation in startup class.
- It indicates that it is a spring context file.

ANNOTATIONS

1. Rest Controller

The @RestController annotation is used to define the RESTful web services. It serves JSON, XML and custom response.

2.Request Mapping

The @RequestMapping annotation is used to define the Request URI to access the REST endpoints.

3.Request Body

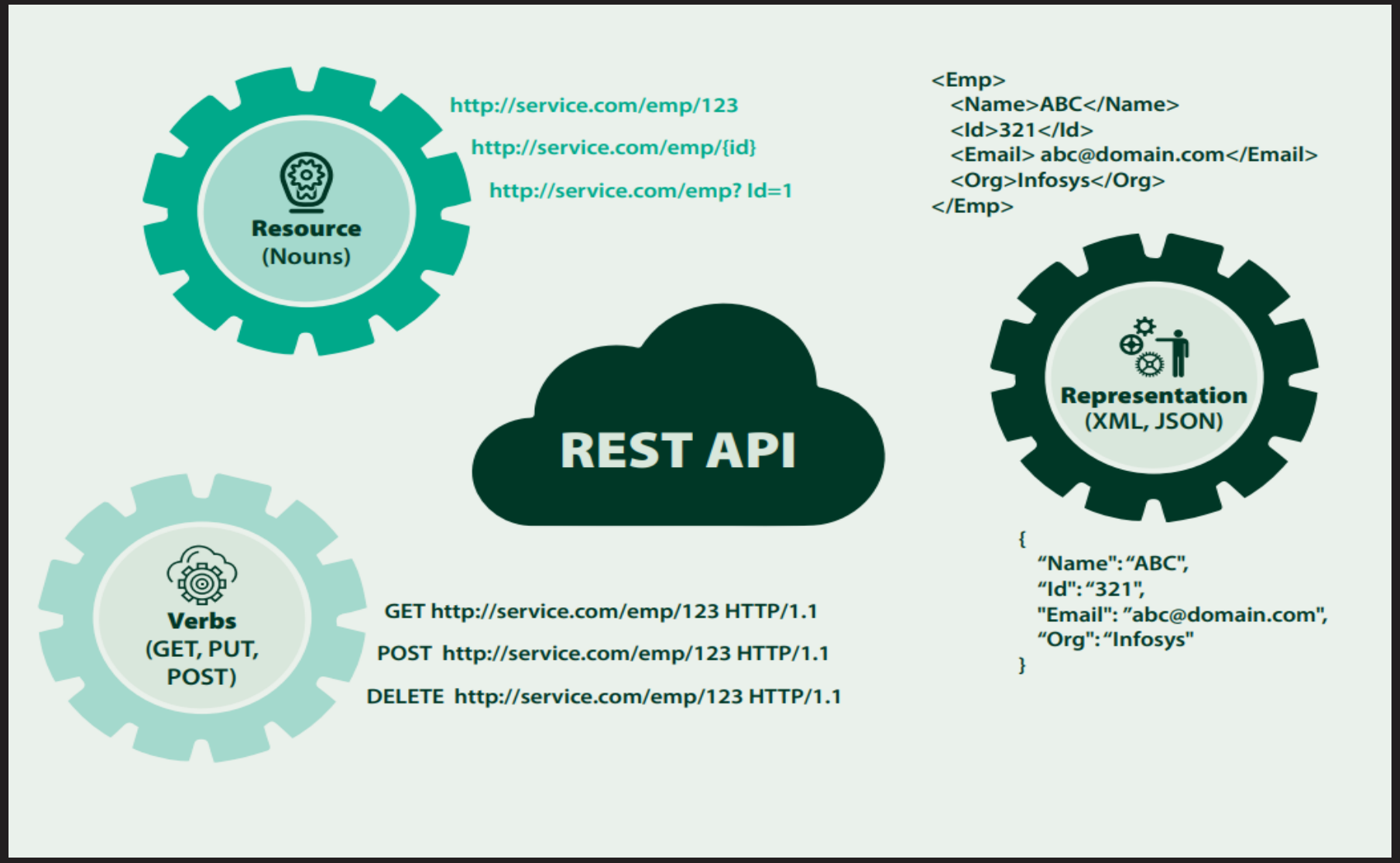
The @RequestBody annotation is used to define the request body content type.

4. Path Variable

The @PathVariable annotation is used to define the custom or dynamic request URI.

Best practices for REST API design

- Accept and respond with JSON
- Use nouns instead of verbs in endpoint paths
- Name collections with plural nouns
- Nesting resources for hierarchical objects
- Handle errors gracefully and return standard error codes
- Allow filtering, sorting, and pagination
- Maintain Good Security Practices
- Cache data to improve performance



Accept and respond with JSON

Even though some people think REST should only return hypertext (including Roy Fielding who created the term) REST APIs should accept JSON for request payload and also send responses to JSON. JSON is the standard for transferring data. Almost every networked technology can use it: JavaScript has built-in methods to encode and decode JSON either through the Fetch API or another HTTP client. Server-side technologies have libraries that can decode JSON without doing much work.

Use nouns instead of verbs in endpoint paths

We shouldn’t use verbs in our endpoint paths. Instead, we should use the nouns which represent the entity that the endpoint that we’re retrieving or manipulating as the pathname.

Use logical nesting on endpoints

When designing endpoints, it makes sense to group those that contain associated information. That is, if one object can contain another object, you should design the endpoint to reflect that.

For example, if we want an endpoint to get the comments for a news article, we should append the /comments path to the end of the /articles path.

Handle errors gracefully and return standard error codes

To eliminate confusion for API users when an error occurs, we should handle errors gracefully and return HTTP response codes that indicate what kind of error occurred.

Common error HTTP status codes include:

400 Bad Request – This means that client-side input fails validation.

401 Unauthorized – This means the user isn't not authorized to access a resource. It usually returns when the user isn't authenticated.

403 Forbidden – This means the user is authenticated, but it's not allowed to access a resource.

404 Not Found – This indicates that a resource is not found.

500 Internal server error – This is a generic server error. It probably shouldn't be thrown explicitly.

Allow filtering, sorting, and pagination

The databases behind a REST API can get very large. Sometimes, there's so much data that it shouldn't be returned all at once because it's way too slow or will bring down our systems. Therefore, we need ways to filter items.

Maintain good security practices

Most communication between client and server should be private since we often send and receive private information. Therefore, using SSL/TLS for security is a must.

A SSL certificate isn't too difficult to load onto a server and the cost is free or very low. There's no reason not to make our REST APIs communicate over secure channels instead of in the open.

Cache data to improve performance

We can add caching to return data from the local memory cache instead of querying the database to get the data every time we want to retrieve some data that users request. The good thing about caching is that users can get data faster

REST VS SOAP

REST is a set of guidelines that offers flexible implementation, whereas SOAP is a protocol with specific requirements like XML messaging.
REST APIs are lightweight, making them ideal for newer contexts like the Internet of Things (IoT), mobile application development, and serverless computing.

Rest should be chosen when we have to develop a highly secure and complex API, which supports different protocols.

Advantages of RESTful web services:

1.Speed: As there is no strict specification, RESTful web services are faster as compared to SOAP. It also consumes fewer resources and bandwidth.

2.Compatible with SOAP: RESTful web services are compatible with SOAP, which can be used as the implementation.

3.Language and Platform Independency: RESTful web services can be written in any programming language and can be used on any platform.

4.Supports Various Data Formats: It permits the use of several data formats like HTML, XML, Plain Text, JSON, etc.

Disadvantages of Restful Web Services

Although REST services tend to provide multiple benefits, still it has given demerits:

•To implement state related query the REST Headers are required which is a clumsy work

•The PUT and DELETE operations are not usable through firewalls or in some browsers.

Conclusion

- RESTful web services are a popular architectural style for building web applications that are scalable, flexible, and easy to maintain.
- By designing web services through RESTful guidelines and best practices, our application can best utilize the in-built features of a web platform and the HTTP protocol.
- The most important takeaways for designing high-quality REST APIs is to have consistency by following web standards and conventions. JSON, SSL/TLS, and HTTP status codes are all standard building blocks of the modern web.
- Developers can enhance productivity and develop loosely coupled web services by adopting the best REST practices