REST API Basics and Key Concepts

By Ramesh Fadatare (Java Guides)

REST Introduction

By Ramesh Fadatare (Java Guides)

By Ramesh Fadatare (Java Guides)

What is REST ?

The REST stands for

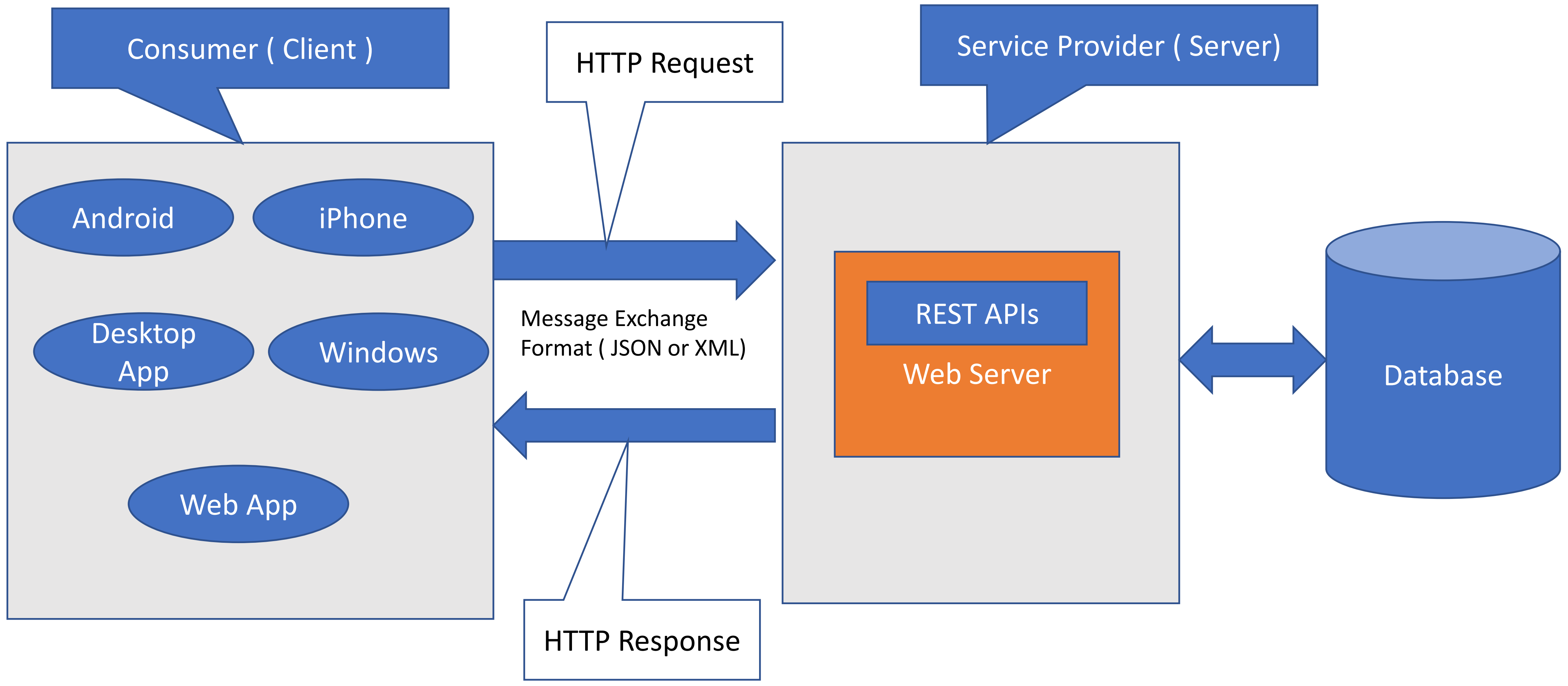
REpresentational

State

Transfer

- **S**tate means data
- **RE**presentational means formats (such as xml, json, yaml, html, etc)
- **T**ransfer means carry data between consumer and provider using HTTP protocol

REST – Architecture



REST - REpresentational State Transfer

- REST was originally coined by Roy Fielding, who was also the inventor of the HTTP protocol.
- A REST API is an intermediary Application Programming Interface that enables two applications to communicate with each other over HTTP, much like how servers communicate to browsers.
- The REST architectural style has quickly become very popular over the world for designing and architecting applications that can communicate.
- The need for REST APIs increased a lot with the drastic increase of mobile devices. It became logical to build REST APIs and let the web and mobile clients consume the API instead of developing separate applications.

REST Architectural Constraints

An API that has following constraints is known as RESTful API:

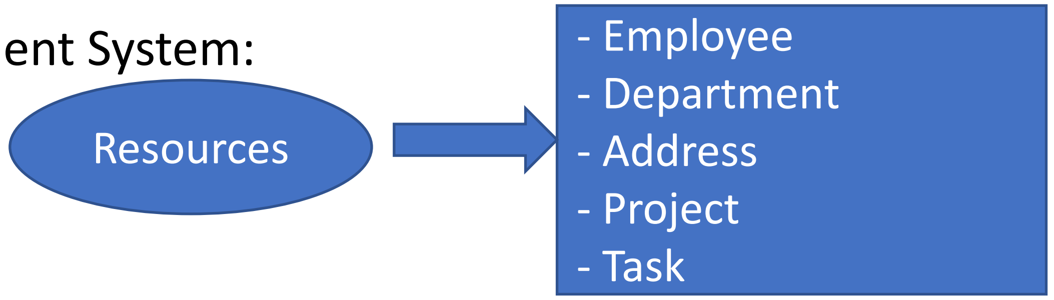
- **Client-server architecture:** The client is the front-end and the server is the back-end of the service. It is important to note that both of these entities are independent of each other.
- **Stateless:** No data should be stored on the server during the processing of the request transfer. The state of the session should be saved at the client's end.
- **Cacheable:** The client should have the ability to store responses in a cache. This greatly improves the performance of the API.
- **Uniform Interface:** This constraint indicates a generic interface to manage all the interactions between the client and server in a unified way, which simplifies and decouples the architecture.
- **Layered System:** The server can have multiple layers for implementation. This layered architecture helps to improve scalability by enabling load balancing.
- **Code on Demand:** This constraint is optional. This constraint indicates that the functionality of the client applications can be extended at runtime by allowing a code download from the server and executing the code.

REST Key Concepts

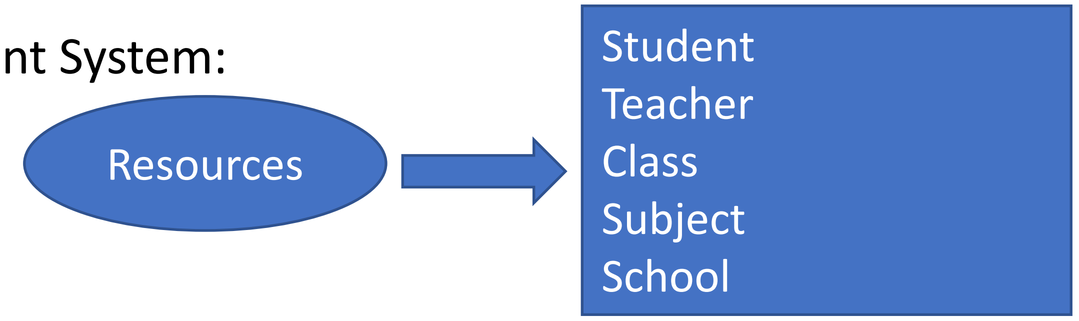
- Resource
- Sub-resource
- URI
- HTTP Methods
- HTTP Status Codes

REST - Resource

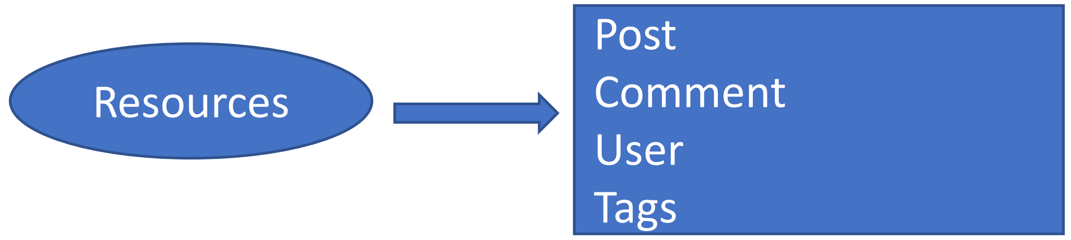
- The fundamental concept of a REST-based system is the resource. A resource is anything you want to expose to the outside world, through your application.
- Employee Management System:



- Student Management System:



- Blog Application



URI - Uniform Resource Identifier

- The resource an be identified by a **Uniform Resource Identifier (URI**). For web-based systems, HTTP is the most commonly used protocol for communicating with external systems. You can identify a unique resource using a URI.
- Consider, we are developing a simple blog application and you can define URIs for a blog Post resource:

GET—http://localhost:8080/api/posts/: Returns a list of all posts
GET—http://localhost:8080/api/posts/2: Returns a post whose ID is 2
POST—http://localhost:8080/api/posts/: Creates a new Post resource
PUT—http://localhost:8080/api/posts/2: Updates a POST resource whose ID is 2
DELETE—http://localhost:8080/api/posts/2: Deletes a POST resource whose ID is 2

REST - Sub-resource

- In REST, the relationships are often modeled by a sub-resource. Use the following pattern for sub-resources.

```
GET  /{resource}/{resource-id}/{sub-resource}
GET  /{resource}/{resource-id}/{sub-resource}/{sub-resource-id}
POST /{resource}/{resource-id}/{sub-resource}
```

```
GET  /{post}/{post-id}/{comments}
GET  /{post}/{post-id}/{comments}/{comment-id}
POST /{post}/{post-id}/{comments}
```

- Use sub-resources child object cannot exist without its parent.

HTTP Methods

HTTP Method	Description	Example
GET	To get a collection or a single resource	http://localhost:8080/api/users http://localhost:8080/api/users/1
POST	To create a new resource	http://localhost:8080/api/users
PUT	To update an existing resource	http://localhost:8080/api/users/1
DELETE	To delete a collection or a single resource	http://localhost:8080/api/users/1

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

HTTP Status Codes

- Some of the frequently used status codes in this class are as follows:
- **200 OK:** This code indicates that the request is successful and the response content is returned to the client as appropriate.
- **201 Created:** This code indicates that the request is successful and a new resource is created.
- **400 Bad Request:** This code indicates that the server failed to process the request because of the malformed syntax in the request. The client can try again after correcting the request.
- **401 Unauthorized:** This code indicates that authentication is required for the resource. The client can try again with appropriate authentication.
- **403 Forbidden:** This code indicates that the server is refusing to respond to the request even if the request is valid. The reason will be listed in the body content if the request is not a HEAD method.
- **404 Not Found:** This code indicates that the requested resource is not found at the location specified in the request.
- **500 Internal Server Error:** This code indicates a generic error message, and it tells that an unexpected error occurred on the server and that the request cannot be fulfilled.