**Week 8**

**What is an Application Programming Interface (API)?**

**Application Programming Interface (API)** is a software interface that allows two applications or software’s to interact, communicate and exchange data with each other without any user intervention.

API is a collection of software functions and procedures.

**How API works?**

* An API is a set of defined rules that explain how computers or applications communicate with one another.
* APIs sit between an application and the web server, acting as an intermediary layer that processes data transfer between systems.
* **Here’s how an API works:**

1. **A client application initiates an API call**to retrieve information—also known as a request. This request is processed from an application to the web server via the API’s Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.
2. **After receiving a valid request**, the API makes a call to the external program or web server.
3. **The server sends a response**to the API with the requested information.
4. **The API transfers the data**to the initial requesting application.

**Why we need APIs?**

* API helps two different software’s to communicate and exchange data with each other.
* It helps you to embed content from any site or application more efficiently.
* APIs can access app components. The delivery of services and information is more flexible.
* Content generated can be published automatically.
* It allows the user or a company to customize the content and services which they use the most.
* Software needs to change over time, and APIs help to anticipate changes.

**API types (Open APIs, Partner APIs, Internal APIs, Composite APIs)**

There are mainly four main types of APIs:

* **Open APIs:** These types of APIs are publicly available to use like OAuth APIs from Google. It has also not given any restriction to use them. So, they are also known as Public APIs.
* **Partner APIs:** Specific rights or licenses to access this type of API because they are not available to the public.
* **Internal APIs:** Internal or private. These APIs are developed by companies to use in their internal systems. It helps you to enhance the productivity of your teams.
* **Composite APIs:** This type of API combines different data and service APIs.

**Types of API Protocols (SOAP, REST)**

# SOAP (Simple Objects Access Protocol)

* It is the oldest API protocol in use, emerging in 1998.
* SOAP uses XML files to transfer data between web services.
* These XML files are sent over HTTP/HTTPS transmissions as is common on the internet.
* SOAP also provides flexibility and enables data transmission over other protocols as well such as Transmission Control Protocol(TCP), Simple Mail Transport Protocol(SMTP), User Data Protocol (UDP), etc.

Messages in SOAP are encoded in XML specifically and have a proper defined format:

1. **Envelope :** Literally envelopes the entire message/data in tags.
2. **Header :** Defines all extra information that might be needed to process this data. This is an optional element.
3. **Body :** This is where we actually write the request for data required/ where all the requested data is added.
4. **Fault :** Defines all the errors that may arise in the data during/due to transmission and measures to handle them.

# REST(Representational State Transfer)

* REST protocols overcome SOAP's dependency on XML by supporting data transmission in multiple formats such as JSON (most prominent), HTML, Python, plain text as well as media files
* REST APIs follow a client-server architecture and must be *stateless*.
* *Stateless* communication implies that no client data is stored between GET requests.
* These GET requests must be distinct and disconnected.REST assigns every operation a unique URL, so when the server receives a request, it knows which instructions to execute to fulfil the request.
* REST also supports caching. So, the browser can store the results obtained from the request locally and retrieve it periodically as needed, thereby increasing speed and efficiency.

A typical REST request has the following components:

1. ***Endpoint* :** The destination URL from which data is being requested.
2. ***Method* :** We use predefined methods such as GET, POST, PUT or DELETE to fetch the data. These methods vary from one other. Ex. in when using GET, the data is appended to the end of the URL string, whereas in POST, the data is sent along with the HTTP request.
3. ***Headers* :** They define the request's details and dictate the proper format in which the response must be received.
4. ***body (data)* :** The actual data sent by the service.

**Common API examples:**

1. Twitter Bots.
2. Log-In Using XYZ.
3. Weather Snippers.
4. Pay with PayPal.
5. Google Maps.
6. Travel Booking.
7. E-Commerce.

**What is API endpoint?**

* An endpoint is one end of a communication channel. When an API interacts with another system, the touchpoints of this communication are considered endpoints. For APIs, an endpoint can include a URL of a server or service.
* Each endpoint is the location from which APIs can access the resources they need to carry out their function.

**Why are API endpoints important?**

* Endpoints help to depict the exact location of the resources to be accessed by API and also play a vital role in ensuring that the software which is interacting with the API is functioning correctly.
* The performance and productivity of APIs depend on its ability to interact and communicate with endpoints effectively.

**API endpoint examples**

* Twitter API Endpoint
* Spotify API Endpoint
* YouTube API Endpoint

**HTTP Concepts**

* HTTP stands for Hypertext Transfer Protocol.
* Hypertext Transfer Protocol is a set of rule which is used for transferring the files like, audio, video, image, text and other multimedia files on the WWW.
* HTTP is an application-level protocol. The communication usually takes place through TCP/IP sockets, but any reliable transport can also be used.
* This latest version is already in use on the web with the help of UDP (User Datagram Protocol) instead of TCP (Transmission Control Protocol) for the underlying transport protocol.
* HTTP is used to make communication between a variety of hosts and clients. It supports a mixture of network configuration.
* HTTP is a protocol that is used to transfer the hypertext from the client end to the server end, but HTTP does not have any security.
* Whenever a user opens their Web Browser, that means the user indirectly uses HTTP

**HTTP working**

* As a request-response protocol, HTTP gives users a way to interact with web resources such as HTML files by transmitting hypertext messages between clients and servers. HTTP clients generally use [Transmission Control Protocol (TCP)](https://www.extrahop.com/resources/protocols/tcp/) connections to communicate with servers.

**HTTP Method (GET, POST, PUT, DELETE)**

1. **GET** requests a specific resource in its entirety
2. **POST** adds content, messages, or data to a new page under an existing web resource
3. **PUT** directly modifies an existing web resource or creates a new URI if need be
4. **DELETE** gets rid of a specified resource

Understanding of JSON structure for API request and response data

**Limitations of JDBC API**

* Since a **PreparedStatement** object represents only one SQL statement at a time, we can execute only one statement by one prepared statement object.
* To prevent injection attacks it does not allow more than one value to a place holder.

**Object relational Mapping – features and benefits**

* ORM stands for **object-relational mapping**, where objects are used to connect the programming language on to the database systems, with the facility to work SQL and object-oriented programming concepts.

**Features:**

* **Idiomatic persistence** : It enables you to write the persistence classes using object oriented classes.
* **High Performance** : It has many fetching techniques and hopeful locking techniques.
* **Reliable** : It is highly stable and eminent. Used by many industrial programmers.

**Benefits:**

* It speeds up development time for teams.
* Decreases the cost of development.
* Handles the logic required to interact with databases.
* Improves security. ORM tools are built to eliminate the possibility of SQL injection attacks.
* You write less code when using ORM tools than with SQL.

**JPA – Java Persistent API**

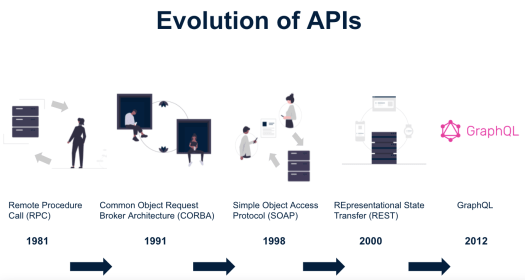
* The Java Persistence API (JPA) is a specification of Java. It is used to persist data between Java object and relational database.
* JPA acts as a bridge between object-oriented domain models and relational database systems.
* As JPA is just a specification, it doesn't perform any operation by itself. It requires an implementation.
* So, ORM tools like Hibernate, TopLink and iBatis implements JPA specifications for data persistence.

**Spring REST controller**

Spring RestController annotation is **used to create RESTful web services using Spring MVC**. Spring RestController takes care of mapping request data to the defined request handler method.

**Basics of REST**

**Evolution of API**

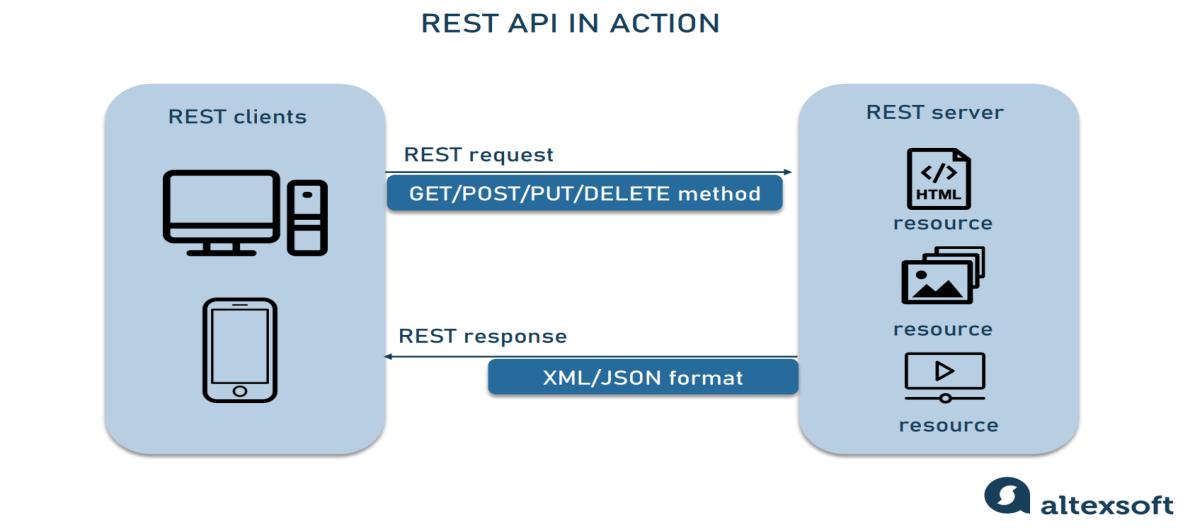


**Overview of REST**

* **R**epresentational **S**tate **T**ransfer (REST) is an architectural style that defines a set of constraints(rules) to be used for creating web services.
* **REST API** is a way of accessing web services in a simple and flexible way without having any processing.
* REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology, simple and flexible making it more suitable for internet usage.
* It’s used to fetch or give some information from a web service. All communication done via REST API uses only HTTP request.

**Working**

* A request is sent from client to server in the form of a web URL as HTTP GET or POST or PUT or DELETE request.
* After that, a response comes back from the server in the form of a resource which can be anything like HTML, XML, Image, or JSON.
* But now JSON is the most popular format being used in Web Services.



**HTTP methods**

Following four HTTP methods are commonly used in REST based architecture.

* GET − Provides a read only access to a resource.
* POST − Used to create a new resource.
* DELETE − Used to remove a resource.
* PUT − Used to update a existing resource or create a new resource.

**REST architectural style**

* REST is a software architectural style that defines the set of rules to be used for creating web services. Web services which follow the REST architectural style are known as RESTful web services. It allows requesting systems to access and manipulate web resources by using a uniform and predefined set of rules.
* Interaction in REST based systems happen through Internet’s Hypertext Transfer Protocol (HTTP).
* A Restful system consists of a
  + client who requests for the resources.
  + server who has the resources.

**Components**

* POST—Creates a resource.
* GET—Retrieves one or more resources.
* PUT—Updates a resource.
* DELETE—Deletes a resource.

**REST Architectural Views**

**Process View:** Process view illustrates the interaction relationship among components. The interaction is in the form of data flow in a system.

**Connector View**: A connector view of architecture tells on the mechanics of the communication between components.

**Data View**: A data view of architecture tells the application state as information flows through the components.

**Architectural Constraints of RESTful API**

There are six architectural constraints.

* **Uniform Interface**: uniform way of interacting with a given server
* **Stateless:**  server would not store anything related to the session. In REST, the client must include all information for the server to fulfill the request.
* **Cacheable:** Every response should include whether the response is cacheable or not and for how much duration responses can be cached at the client side. Client will return the data from its cache for any subsequent request and there would be no need to send the request again to the server.
* **Client-Server:** REST application should have a client-server architecture. A Client is someone who is requesting resources and server is someone who holds the resources.
* **Layered System:** A REST API can consist of multiple layers, eg., business logic, presentation, data access
* **Code on Demand:** It is an optional feature. According to this, servers can also provide executable code to the client.

**Properties of REST API**

1. **Performance:** REST can support the Performance goal by using caches to keep available data close to where it is being processed.

2 **Scalability:** Scalability means to consistently provide service regardless of the increase or decrease of web users.

3. **Simplicity**: By separating the functionality within components in a system, we can induce the simplicity of the architectural styles.

4. **Modifiability:** Modifiability represents the ease at which changes can be incorporated into the architecture.

## 5. Visibility: Visibility refers to the ability of a component to monitor or mediate the interaction between two components.

## 6. Portability: Portability represents the each with which a system can be moved from one deployed location to other. portability of components by moving program code with the data;

## 7. Reliability: An architecture style can improve reliability by avoiding single points of failure, using failover mechanisms, and be relying on monitoring features that can dynamically anticipate and respond to failure conditions.

## REST API Design Principles

## REST APIs are designed around resources, which are any kind of object, data, or service that can be accessed by the client.

## A resource has an identifier*,* which is a URI that uniquely identifies that resource. For example, the URI for a particular customer order might be:

https://adventure-works.com/orders/1

* Clients interact with a service by exchanging representationsof resources. Many web APIs use JSON as the exchange format. For example, a GET request to the URI listed above might return this response body:

{"orderId":1,"orderValue":99.90,"productId":1,"quantity":1}

* REST APIs use a uniform interface, which helps to decouple the client and service implementations. For REST APIs built on HTTP, the uniform interface includes using standard HTTP verbs to perform operations on resources. The most common operations are GET, POST, PUT, PATCH, and DELETE.
* REST APIs use a stateless request model. HTTP requests should be independent and may occur in any order
* REST APIs are driven by hypermedia links that are contained in the representation.

## What is API?

An **Application Programming Interface (API)** establishes a connection between computers or between computer programs (applications) by providing readily available codes and information pipelines. It is a type of software interface that acts as a mediator among other pieces of software to streamline the interaction with one other.

## What is REST API?

Representational State Transfer, also known as **REST**, is basically a standardized Software Architecture Style, or in simple words, a specific type of API used by the industry to establish a connection between Client and Server. REST API is built to guide the development and design of the World Wide Web’s architecture.

REST APIs provide a flexible, lightweight way of integrating computer applications. REST APIs are a simple and standardized approach to communication, which means you don’t have to worry about how to format your data, it’s all standardized and industry use.

**What is Spring Boot RESTful service?**

RESTful Web Services are **client and server applications that communicate over the WWW**. RESTful Web Services are REST Architecture based Web Services. In REST Architecture, everything is a resource. RESTful Web Services provides communication between software applications running on different platforms and frameworks.

**What are examples of RESTful web services?**

**Facebook, Twitter, and Google** expose their functionality in the form of Restful web services.

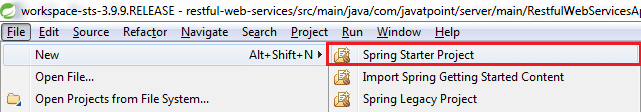
**How to create RESTful service**

**Step 1:** Download the **Spring Tool Suite (STS)** from [https://spring.io/tools3/sts/all](https://spring.io/tools3/sts/all" \t "_blank)

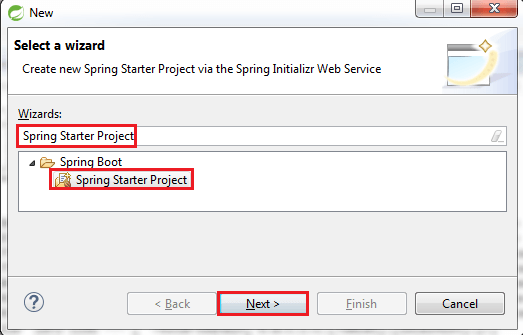
and extract it.

**Step 2:** Launch the **STS**.

**Step 3:** Click on **File menu -> New -> Spring Starter Project ->**



If the **Spring Starter Project** is not enlisted, then click on **Other** at the bottom of the menu. A dialog box appears on the screen. Type **Spring Starter Project** in the **Wizards** text box and click on the **Next** button.



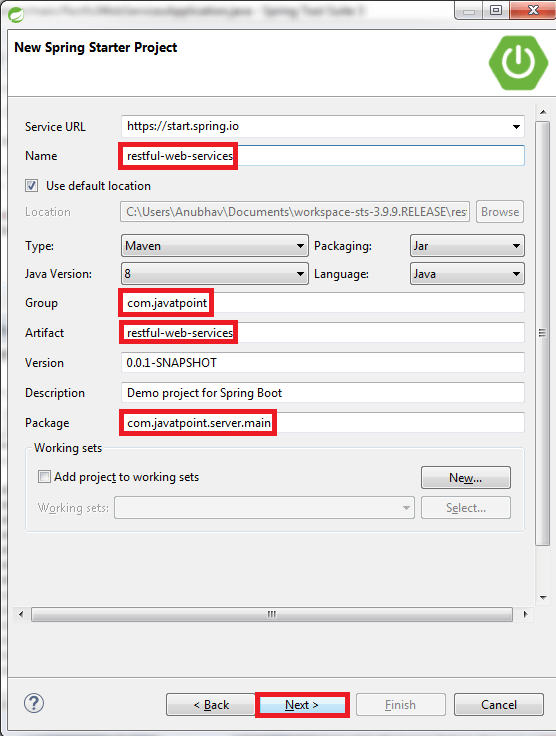
**Step 4:** provide the name, group, and package of the project. We have provided:

Name: **restful-web-services**

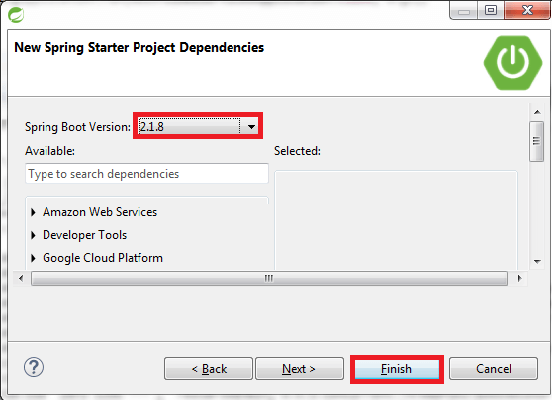
Group: **com.javatpoint**

Package: **com.javatpoint.server.main**

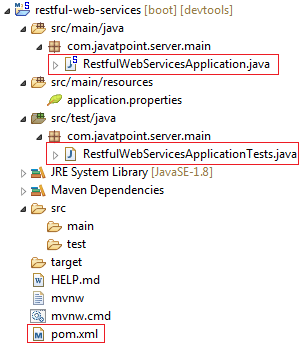
Click on the **Next** button.



**Step 5:** Choose the Spring Boot Version **2.1.8** and add **Spring Web MVC, Spring Boot DevTools, JPA,** and **H2** dependencies in the pom.xml.



**Step 6:** We can see the project structure in the project explorer window.



**Step 7:** Now open the **RestfulWebServicesApplication.java** file and Run the file as Java Application.

## Creating a Hello World Service

**Step 1:** Create a new class with the name **HelloWorldController** in the package **com.javatpoint.server.main**.

**Step 2:** Whenever we create a web service, we need to define two things **Get** method and the **URI**. Now create the **helloWorld()** method which returns the string "Hello World." If we want to tell the spring MVC that it is going to handle the REST request, we have to add **@RestController** annotation. Now it becomes a rest controller which can handle the Rest request.

**package** com.javatpoint.server.main;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RestController;

//Controller

@RestController

**public** **class** HelloWorldController

{

//using get method and hello-world as URI

@GetMapping(path="/hello-world")

**public** String helloWorld()

{

**return** "Hello World";

}

}

**Step 3:** Run the **RestfulWebServiceApplication**. It displays the string **Hello World** on the browser.

## Enhancing the Hello World Service to Return a Bean

In this section, we are going to generate a bean for the method helloWorld().

**Step 1:** Create a **helloWorldBean()** method in **HelloWordController.java** file. Map the URI to "**/hello-world-bean**" and return **HelloWorldBean**.

**HelloWorldController.java**

**package** com.javatpoint.server.main;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RestController;

//Controller

@RestController

**public** **class** HelloWorldController

{

//using get method and hello-world URI

@GetMapping(path="/hello-world")

**public** String helloWorld()

{

**return** "Hello World";

}

@GetMapping(path="/hello-world-bean")

**public** HelloWorldBean helloWorldBean()

{

**return** **new** HelloWorldBean("Hello World"); //constructor of HelloWorldBean

}

}

**Step 2:** Create a class **HelloWorldBean**.

**Step 3:** Generate Getters and **Setters**.

Right-click -> Source -> Generate Getters and Setters -> check the box -> Ok

**Step 4:** Generate **toString()**..

Right-click -> Source -> Generate toString().. -> Ok

**package** com.javatpoint.server.main;

**public** **class** HelloWorldBean

{

**public** String message;

//constructor of HelloWorldBean

**public** HelloWorldBean(String message)

{

**this**.message=message;

}

//generating getters and setters

**public** String getMessage()

{

**return** message;

}

**public** **void** setMessage(String message)

{

**this**.message = message;

}

@Override

//generate toString

**public** String toString()

{

**return** String.format ("HelloWorldBean [message=%s]", message);

}

}

**Step 5:** Launch the **HelloWorldController**. The URL of the browser changesto **localhost:8080/hello-world-bean**.

It returns the message "**Hello World**" in JSON format.

**Output:**

{  message: "Hello World"  }