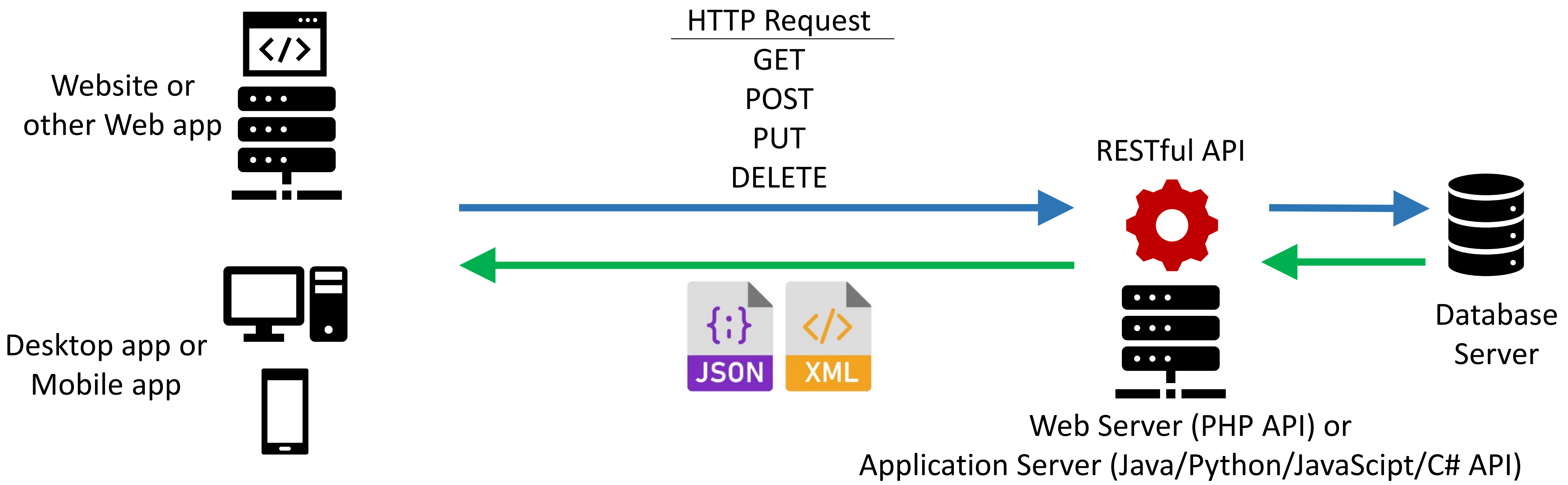
Internet Technologies

RESTful API Client

University of Cyprus
Department of Computer Science

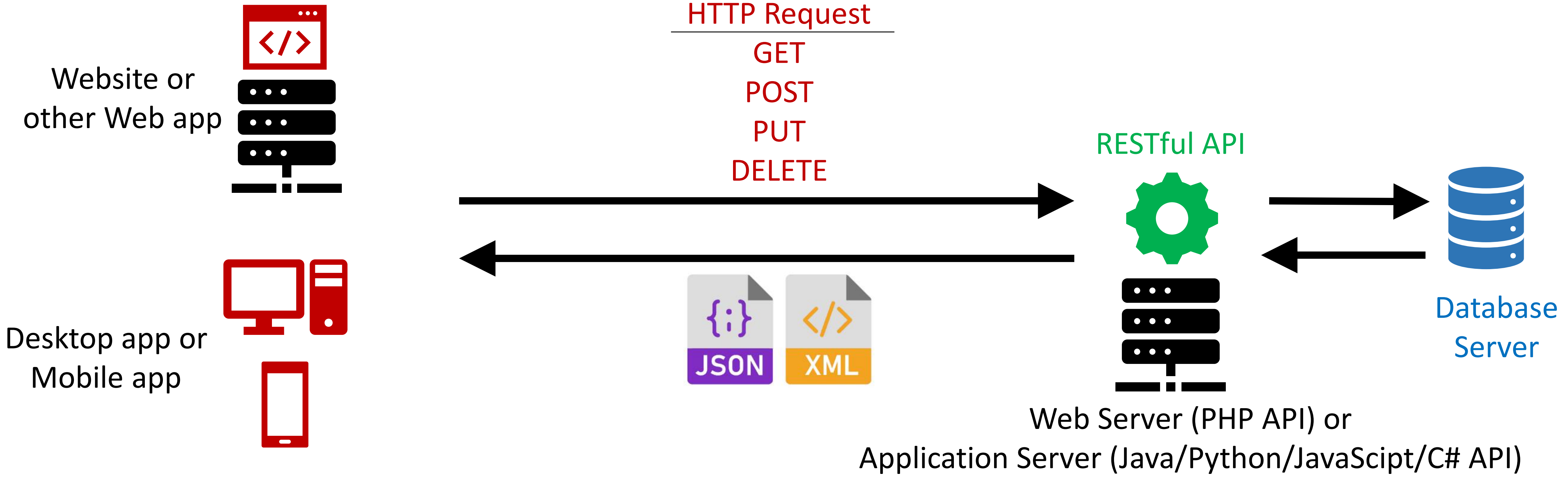
What is RESTful API?

- RESTful API is an interface between software components (e.g. apps and databases) communicating over the Internet that want to exchange (upload / download) data



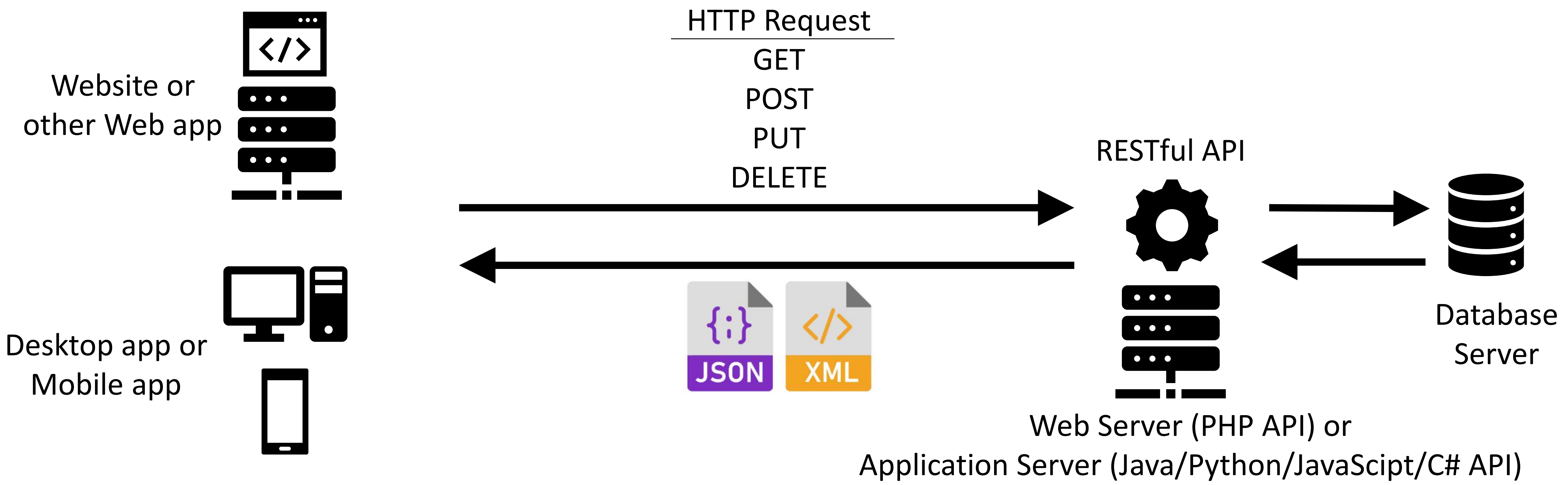
What is RESTful API?

- Apps send HTTP messages (e.g. GET *to download data*, POST *to upload data*) to RESTful API
- RESTful API pushes/pulls data to/from database and replies to apps appropriately



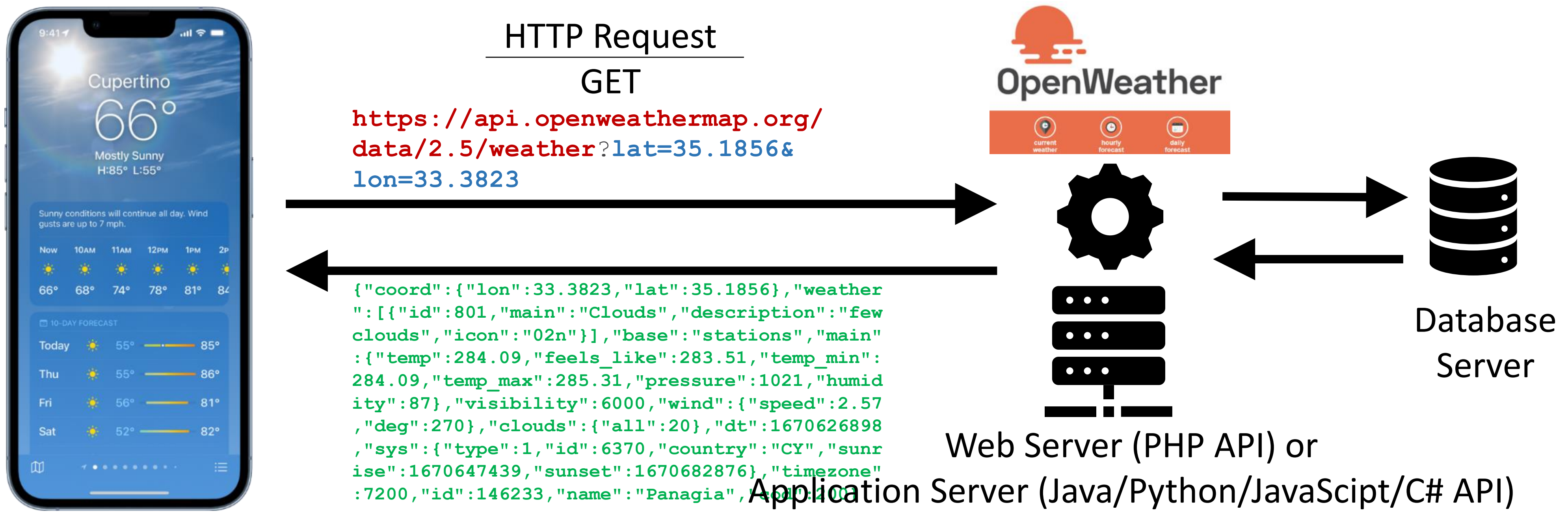
What is RESTful API?

- Exchanged data is described using specific formats such as JSON and XML (JSON is more popular because it is more lightweight & easier to parse)


Website or other Web app
Desktop app or Mobile app
HTTP Request
GET
POST
PUT
DELETE
JSON
XML
RESTful API
Web Server (PHP API) or Application Server (Java/Python/JavaScipt/C# API)
Database Server

What is RESTful API?

- Example: a mobile application calls a Weather API function (endpoint) to get the current weather conditions of a given location. Weather API queries the database, formats query results as JSON and sends it back to app
- App extracts data from JSON to display on its UI

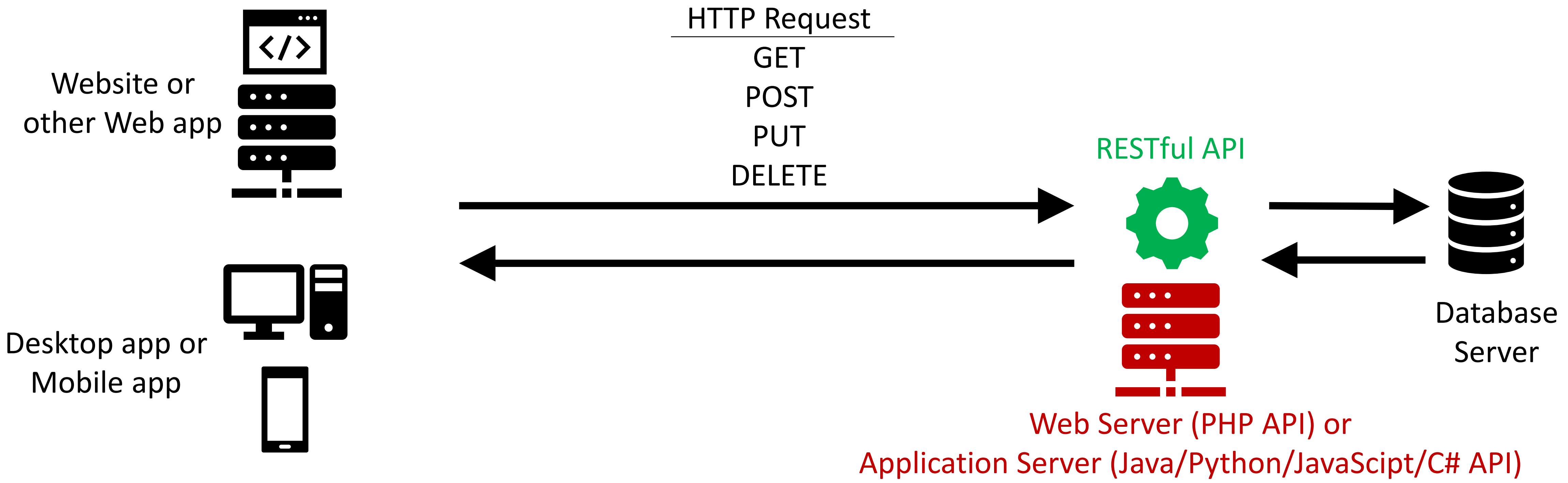


```
https://api.openweathermap.org/
data/2.5/weather?lat=35.1856&
lon=33.3823
```

{"coord":{"lon":33.3823,"lat":35.1856},"weather":[{"id":801,"main":"Clouds","description":"few clouds","icon":"02n"}],"base":"stations","main":{"temp":284.09,"feels_like":283.51,"temp_min":284.09,"temp_max":285.31,"pressure":1021,"humidity":87},"visibility":6000,"wind":{"speed":2.57,"deg":270},"clouds":{"all":20},"dt":1670626898,"sys":{"type":1,"id":6370,"country":"CY","sunrise":1670647439,"sunset":1670682876},"timezone":7200,"id":146233,"name":"Panagia","cod":200}

What is RESTful API?

- RESTful APIs can be built with server-side programming languages such as Java, Python, JavaScript, C# (hosted on application servers) or PHP (hosted on web servers)

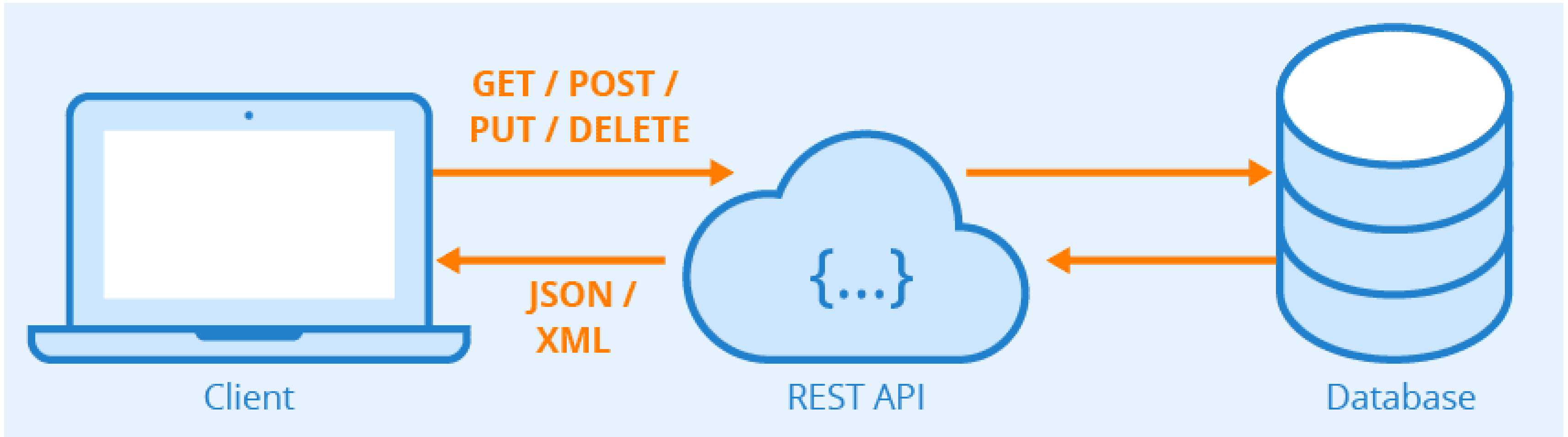


RESTful API Usage

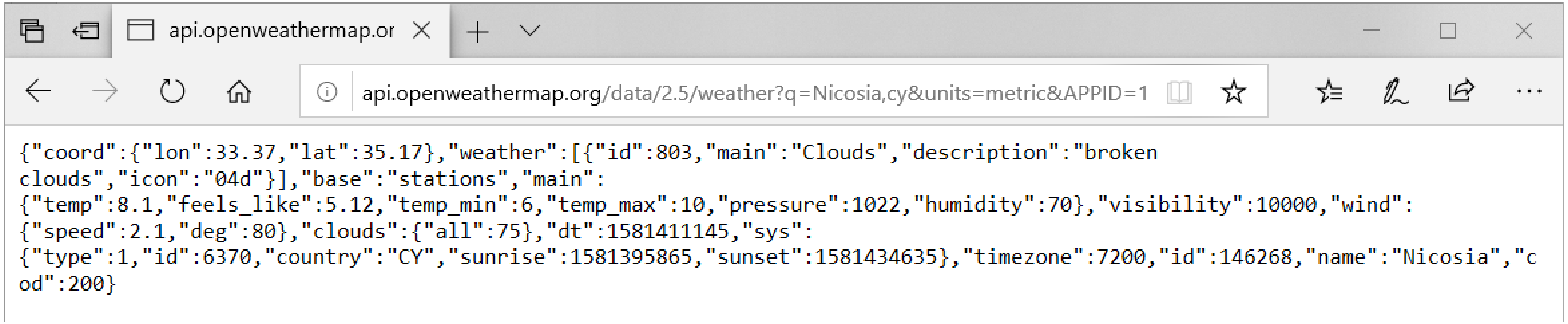
- Over the past few years, RESTful APIs have gained popularity in the market
    - For instance, each time you check the weather or book a travel ticket, one or more APIs are involved for pulling data from databases
- RESTful APIs enable businesses to open their applications' data and functionality to external third-party developers, it eventually grows business partnerships, driving more revenue.

4 Commonly Used RESTful API Methods

- Each request is sent as an **HTTP request**
  - **GET:** Receive information about an API resource
  - **POST:** Create a new API resource
  - **PUT:** Update an existing API resource
  - **DELETE:** Delete an API resource
- Requests are sent to base URL, also known as an "API Endpoint"



RESTful API Endpoint example


api.openweathermap.or
api.openweathermap.org/data/2.5/weather?q=Nicosia,cy&units=metric&APPID=1

```
{"coord":{"lon":33.37,"lat":35.17},"weather":[{"id":803,"main":"Clouds","description":"broken
clouds","icon":"04d"}],"base":"stations","main":
{"temp":8.1,"feels_like":5.12,"temp_min":6,"temp_max":10,"pressure":1022,"humidity":70},"visibility":10000,"wind":
{"speed":2.1,"deg":80},"clouds":{"all":75},"dt":1581411145,"sys":
{"type":1,"id":6370,"country":"CY","sunrise":1581395865,"sunset":1581434635},"timezone":7200,"id":146268,"name":"Nicosia","c
od":200}
```

- api.openweathermap.org/data/2.5/weather?q=Nicosia,cy&units=metric&APPID=xxxx
  - weather: Tells the server that we are requesting the "current weather" resource
  - q=Nicosia,cy&units=metric : Query the server to return weather about Nicosia,cy location in metric system of measurement (Celsius, meters)
  - APPID=xxxx : Tells the server the identifier of the API caller (caller authentication)

parameters

*** OpenWeatherMap API offers weather information for locations across the globe. Create account and get APIID from here.

3rd-Party RESTful APIs

- Many websites expose RESTful APIs to outside developers. These are often called "3rd-party APIs" or "Developer APIs“
- Examples:
  - Spotify
  - Giphy
  - GitHub
  - Google APIs
  - Facebook
  - Instagram
  - etc...

Try Googling "<product name> API" to see if one exists for a given company!

Open API Example: Cyprus Water

- Cyprus Water is an open RESTful API that developers can query to get data and functionality on water reservoirs in Cyprus
- Available endpoints:

  Base URL: `https://cyprus-water.appspot.com`

Method	Endpoint	Usage	Returns	Required parameters
GET	`/api/dams`	Static information about the main water reservoirs	json	
GET	`/api/date-statistics`	Statistics of water reservoirs on a specific date	json	date=[yyyy-MM-dd]
GET	`/api/percentages`	Storage percentages of the main water reservoirs on a specific date	json	date=[yyyy-MM-dd]
GET	`/api/monthy-inflows`	Historical monthly inflows throughout time	json	

How to consume/build RESTful APIs?

- **Consume** RESTful Web Services (RESTful API client)
  - The easiest way to start using an API is by finding a RESTful client application online, like Postman, or Paw (for MAC). These ready-made (and often free) tools help you structure your HTTP requests to consume existing REST APIs
  - Develop JAVA RESTful API client: **Jersey** (https://eclipse-ee4j.github.io/jersey/), **Spring Boot** (https://spring.io/guides/gs/consuming-rest/)
- **Serve** RESTful Web Services (RESTful API server): (NEXT LABs)
  - Develop JAVA RESTful API server: **Jersey** (https://eclipse-ee4j.github.io/jersey/), **Spring Boot** (https://spring.io/guides/gs/rest-service/)
  - Develop JavaScript RESTful API server: **Node.js** (https://nodejs.org/) and **Express** (https://expressjs.com/)
  - Develop Python RESTful API server: **Flask** (https://flask-restful.readthedocs.io/en/latest/), **Django REST framework** (https://www.django-rest-framework.org/)

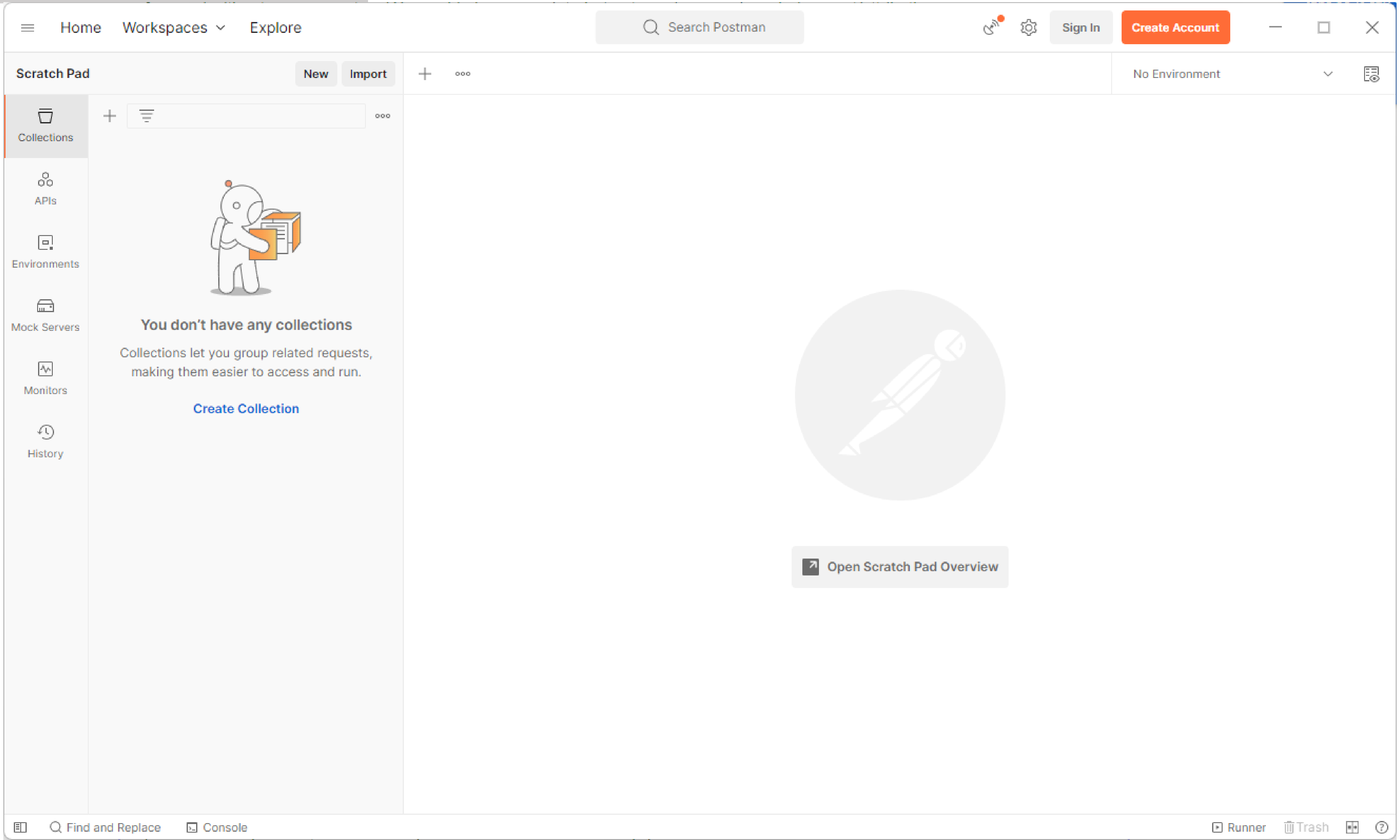
Java or Python for serving RESTful APIs?

- Java is recommended for enterprise-level, high-load APIs
  - Slower development time
  - Heavier resource (RAM) usage
  - Easier application packaging (.jar)
  - Significant version dependence => expensive system support
- JavaScript is recommended for fast-prototyping, medium-load APIs
  - Use the same familiar syntax for both client and server-side tasks (faster development time)
  - Lightweight resource usage, ideal for real-time data processing
  - Slower than Java
- Python is recommended for fast-prototyping, low-load, personal-use APIs
  - Faster development time
  - No compilation, faster testing
  - Minimal version dependence (given than Python 2.x is deprecated and rarely used)

Postman: RESTful API client to generate HTTP requests and receive data

- Complete toolchain for API developers
- Offers a lot of features to simplify generating web server requests
- Used by over 20 million developers worldwide to access million of APIs every month
- Available for Windows, Linux, Mac
- Can downloaded for free from the project website
  - You can install it on your Windows (host machine) as a Windows APP or as a plugin of your web browser
  - Available as a snap package in Ubuntu VM
    - Installation command: `sudo snap install postman`

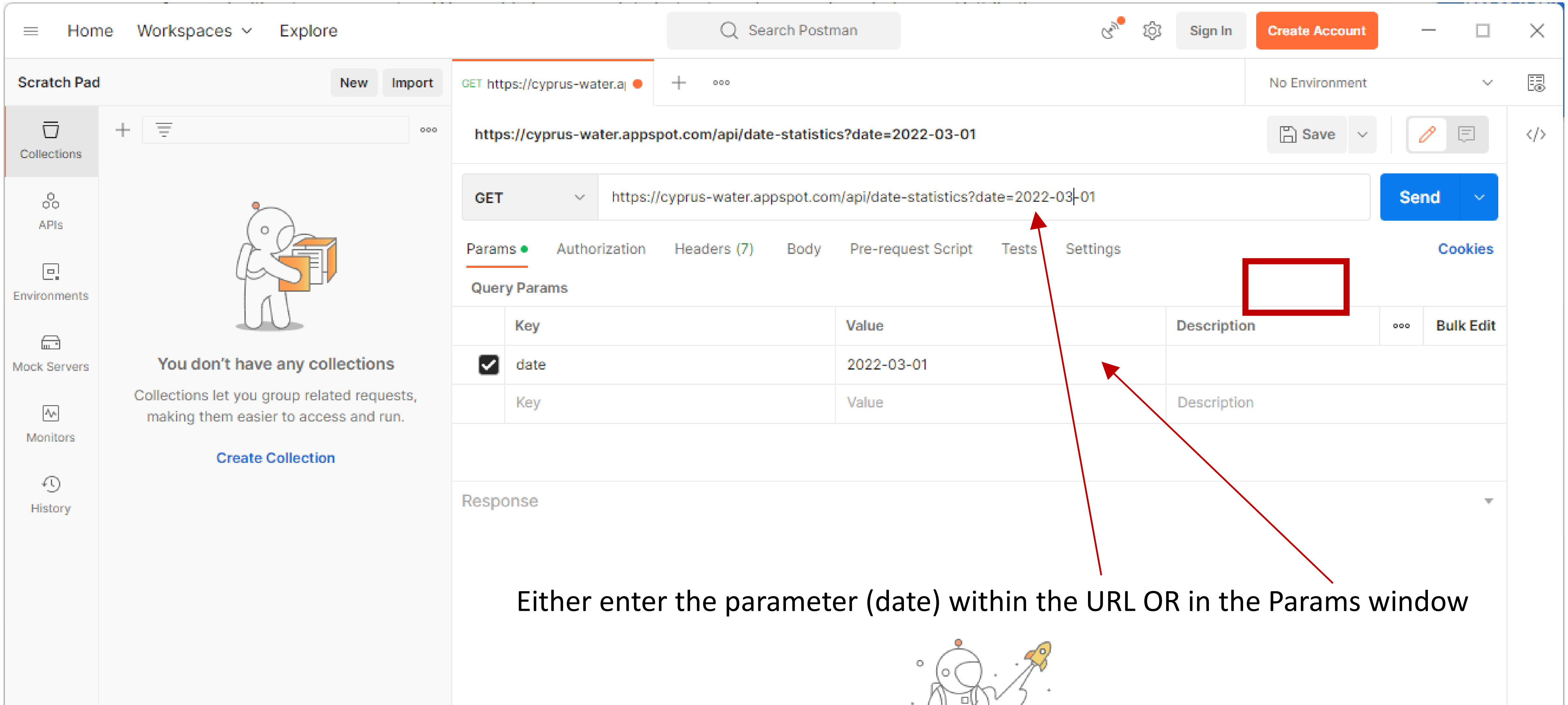
RESTful Client Application: Postman



You don't have any collections

Collections let you group related requests, making them easier to access and run.

Use Postman to send GET message to `/date-statistics` endpoint in Cyprus Water RESTful API



Home Workspaces Explore

Scratch Pad New Import

GET https://cyprus-water.appspot.com/api/date-statistics?date=2022-03-01

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description
✓	date	2022-03-01	
	Key	Value	Description

Response

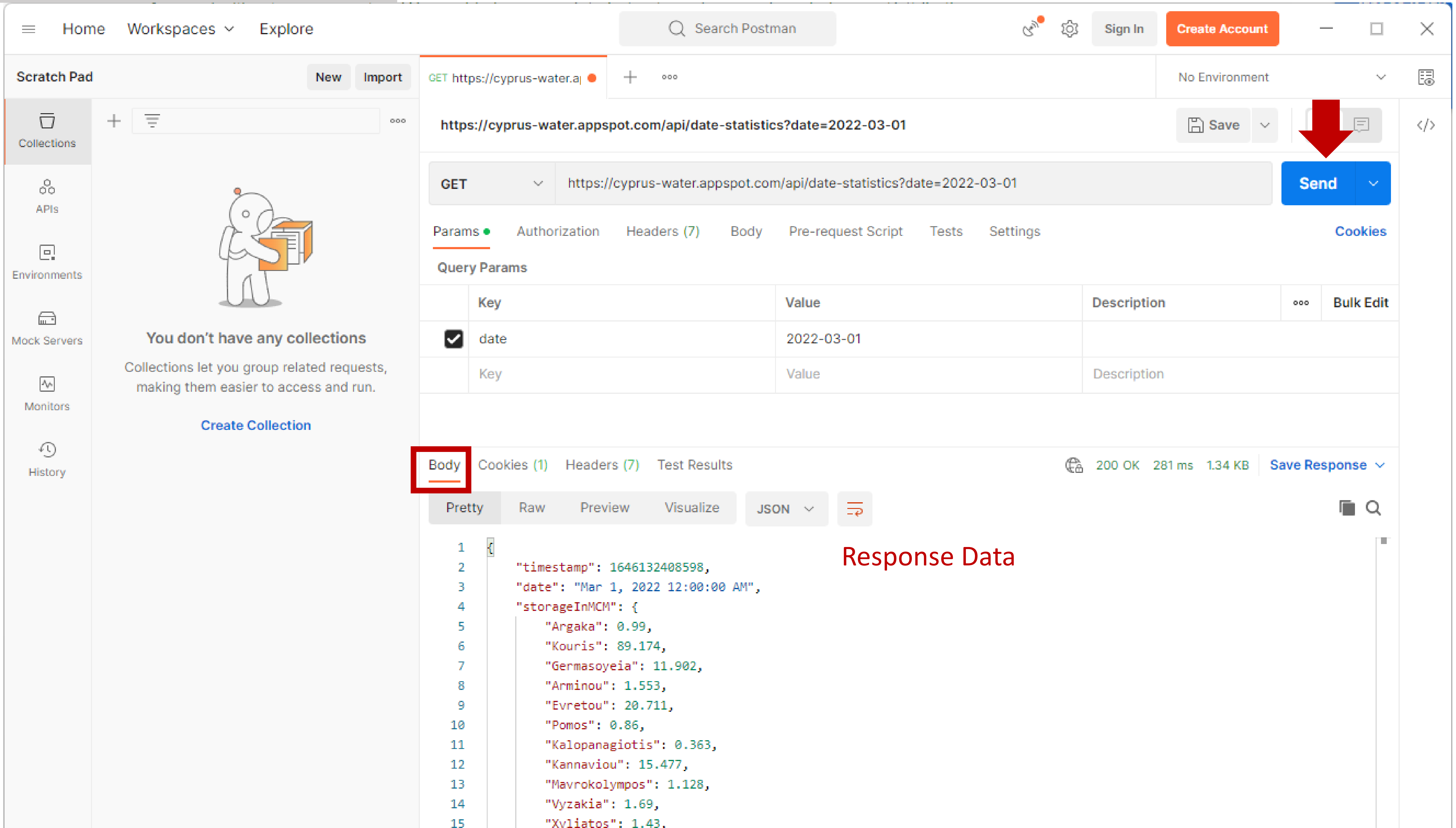
You don't have any collections

Collections let you group related requests, making them easier to access and run.

Create Collection

Either enter the parameter (date) within the URL OR in the Params window

RESTful Client Application: Postman



You don't have any collections

Collections let you group related requests, making them easier to access and run.

GET https://cyprus-water.appspot.com/api/date-statistics?date=2022-03-01

Query Params

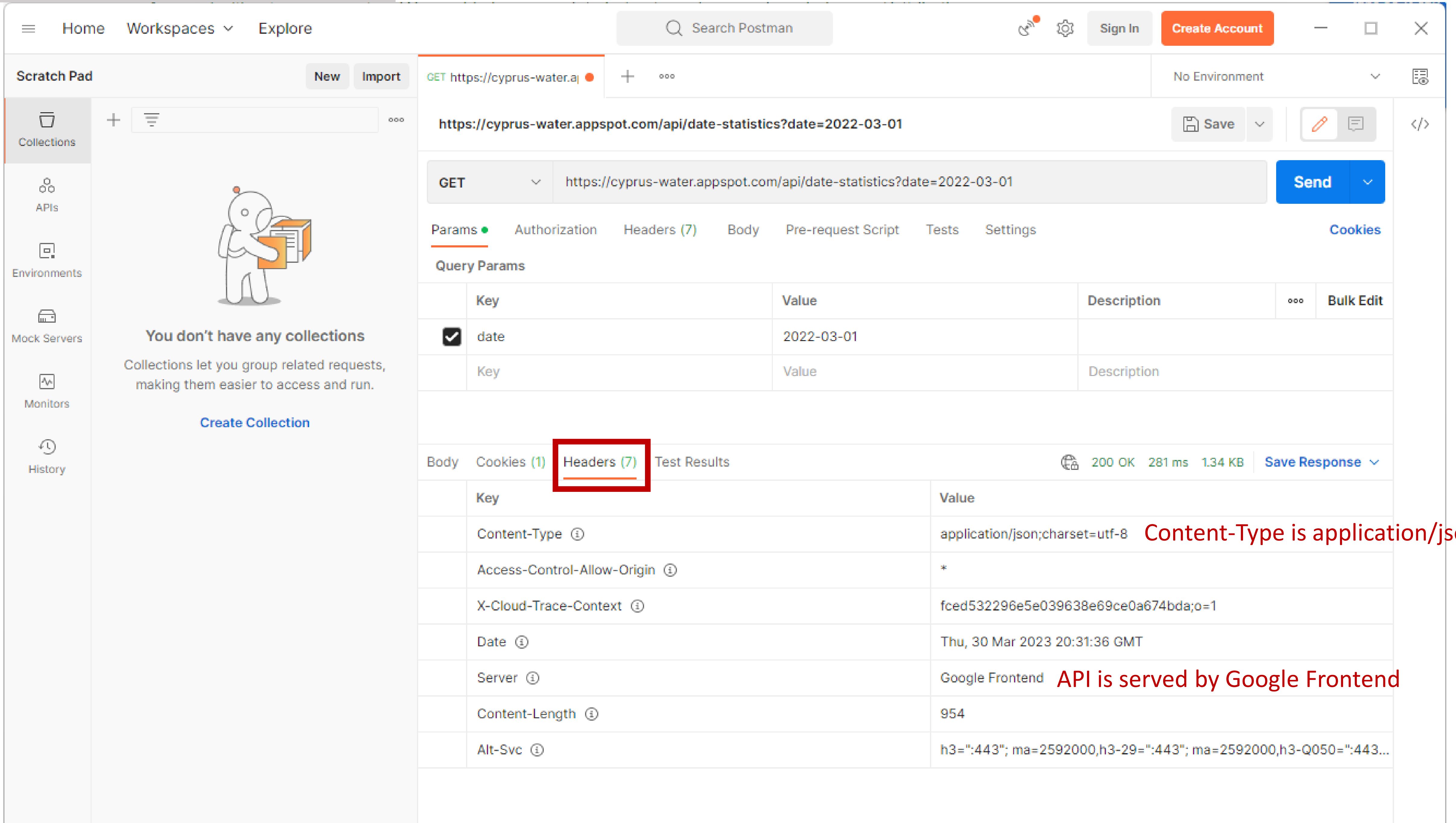
	Key	Value	Description
✓	date	2022-03-01	

200 OK 281 ms 1.34 KB

Response Data

```
{
    "timestamp": 1646132408598,
    "date": "Mar 1, 2022 12:00:00 AM",
    "storageInMCM": {
        "Argaka": 0.99,
        "Kouris": 89.174,
        "Germasoyeia": 11.902,
        "Arminou": 1.553,
        "Evretou": 20.711,
        "Pomos": 0.86,
        "Kalopanagiotis": 0.363,
        "Kannaviou": 15.477,
        "Mavrokolympos": 1.128,
        "Vyzakia": 1.69,
        "Xyliatos": 1.43,
```

RESTful Client Application: Postman



GET https://cyprus-water.appspot.com/api/date-statistics?date=2022-03-01

Query Params

	Key	Value	Description
✓	date	2022-03-01	
	Key	Value	Description

200 OK 281 ms 1.34 KB

Key	Value
Content-Type	application/json;charset=utf-8
Access-Control-Allow-Origin	*
X-Cloud-Trace-Context	fced532296e5e039638e69ce0a674bda;o=1
Date	Thu, 30 Mar 2023 20:31:36 GMT
Server	Google Frontend
Content-Length	954
Alt-Svc	h3=":443"; ma=2592000,h3-29=":443"; ma=2592000,h3-Q050=":443...

Content-Type is application/json

API is served by Google Frontend

Prerequisites

- Download and install Java/JDK (if JDK 17 or later is not already installed on your machine)
- Set the JAVA_HOME environment variable
- Download and install latest Maven package
- Set the MAVEN_HOME environment variable
- Download and Install Python (if Python 3.x or Anaconda is not already installed on your machine)
- Open VS Code and install the following extensions:
  - Project Manager for Java by Microsoft
  - Maven for Java by Microsoft

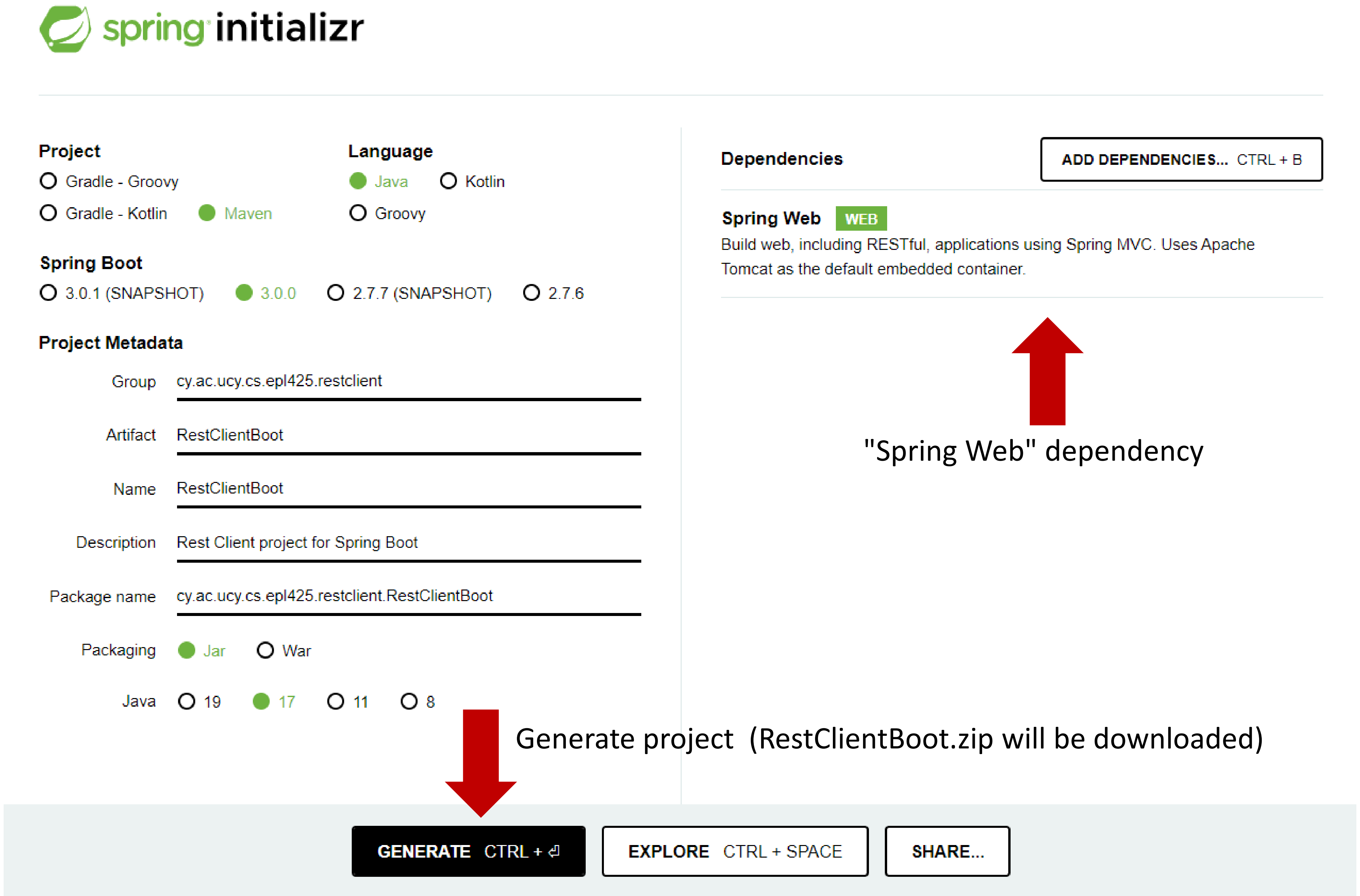
RESTful API Client in Java using Spring Boot

- Spring Boot makes it easy to create stand-alone, production/enterprise-level applications easily that you can "just run"
  - Provides **boilerplate** (pre-written) code (that may be reuse on various projects with little or no modification) to save developers from repeating common steps
- Getting Started
  - Super quick — try the Quickstart Guide.
  - More general — try Building an Application with Spring Boot
  - More specific — try Consuming a RESTful Web Service (REST Client).
  - More specific — try Building a RESTful Web Service (REST Server) – NEXT LAB
  - Or search through all guides on the Guides homepage.

RESTful API Client in Java using Spring Boot

- Build an application that uses Spring's `RestTemplate`
- **Start from scratch**: Spring Initializr
  - Web-based, fast way to pull in all the dependencies we need for an application
  - In this project, we need to include only the "Spring Web" dependency
  - After we set the parameters (see next slide) we press Generate at the bottom of the page to download the zip folder of the project

RESTful Client Development: Spring Boot



spring initializr

Project
- ○ Gradle - Groovy
- ○ Gradle - Kotlin
- ● Maven

Language
- ● Java
- ○ Kotlin
- ○ Groovy

Spring Boot
- ○ 3.0.1 (SNAPSHOT)
- ● 3.0.0
- ○ 2.7.7 (SNAPSHOT)
- ○ 2.7.6

Project Metadata

Group	cy.ac.ucy.cs.epl425.restclient
Artifact	RestClientBoot
Name	RestClientBoot
Description	Rest Client project for Spring Boot
Package name	cy.ac.ucy.cs.epl425.restclient.RestClientBoot
Packaging	● Jar ○ War
Java	○ 19 ● 17 ○ 11 ○ 8

Dependencies

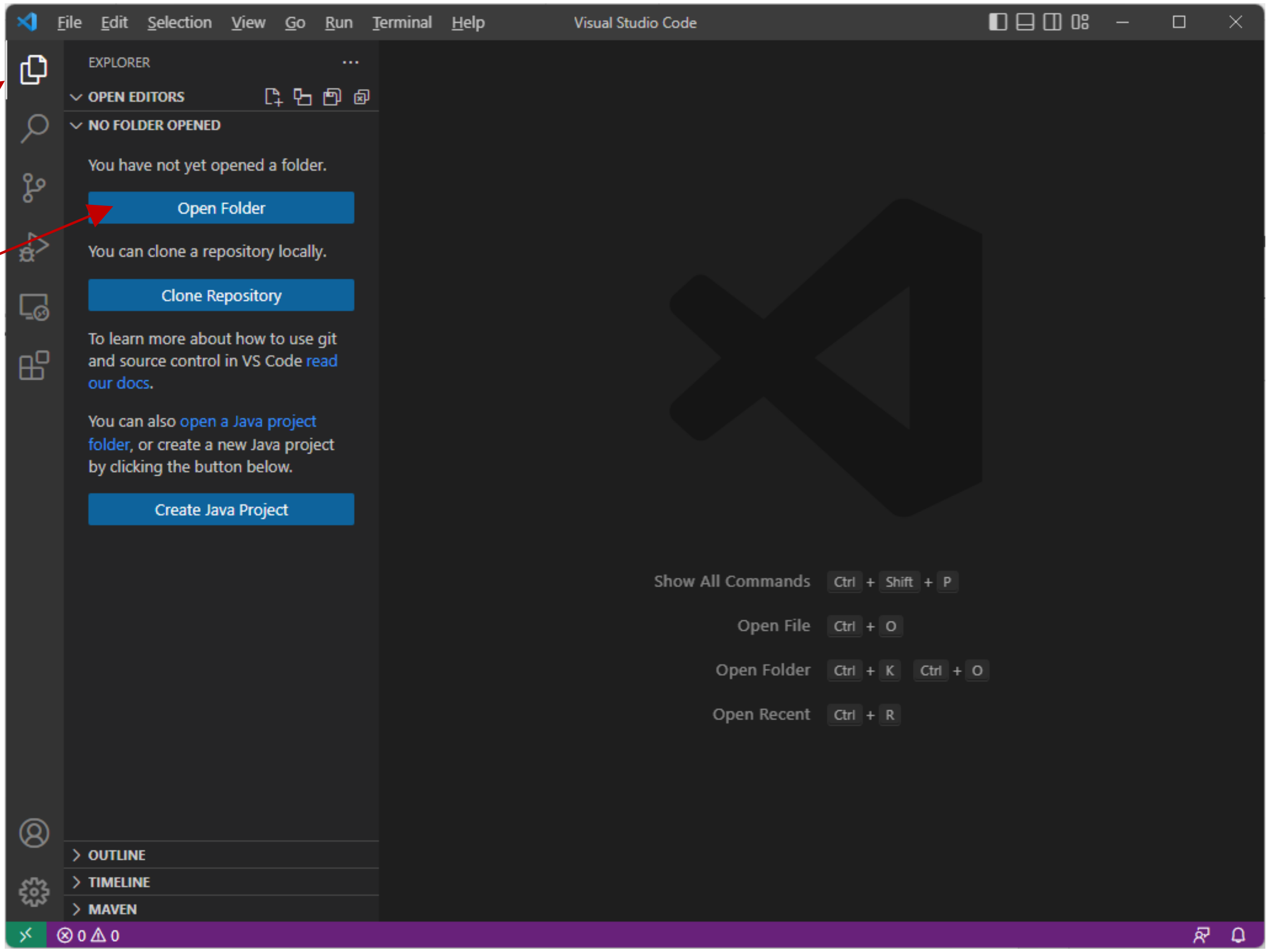
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

"Spring Web" dependency

Generate project (RestClientBoot.zip will be downloaded)

Open Spring Boot Project in VS Code

- Extract RestClientBoot.zip
- Open VS code
- Click on Explorer tab
- Click on Open Folder
- Select the RestClientBoot directory



You have not yet opened a folder.

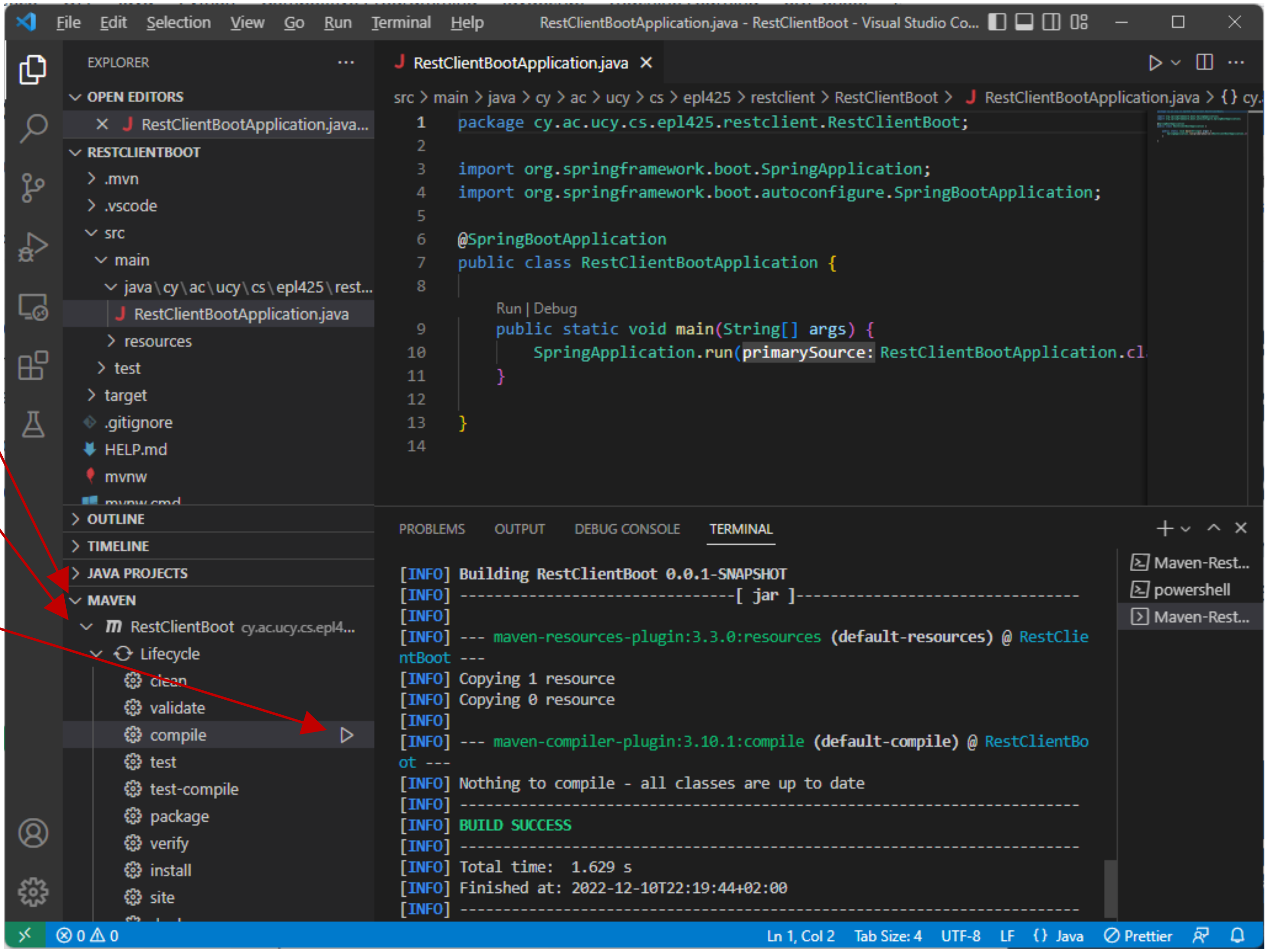
You can clone a repository locally.

To learn more about how to use git and source control in VS Code read our docs.

You can also open a Java project folder, or create a new Java project by clicking the button below.

Compile Spring Boot Maven Project

- Open the MAVEN tab in EXPLORER
- Select the RestClientBoot project
- Open Lifecycle and run the compile command

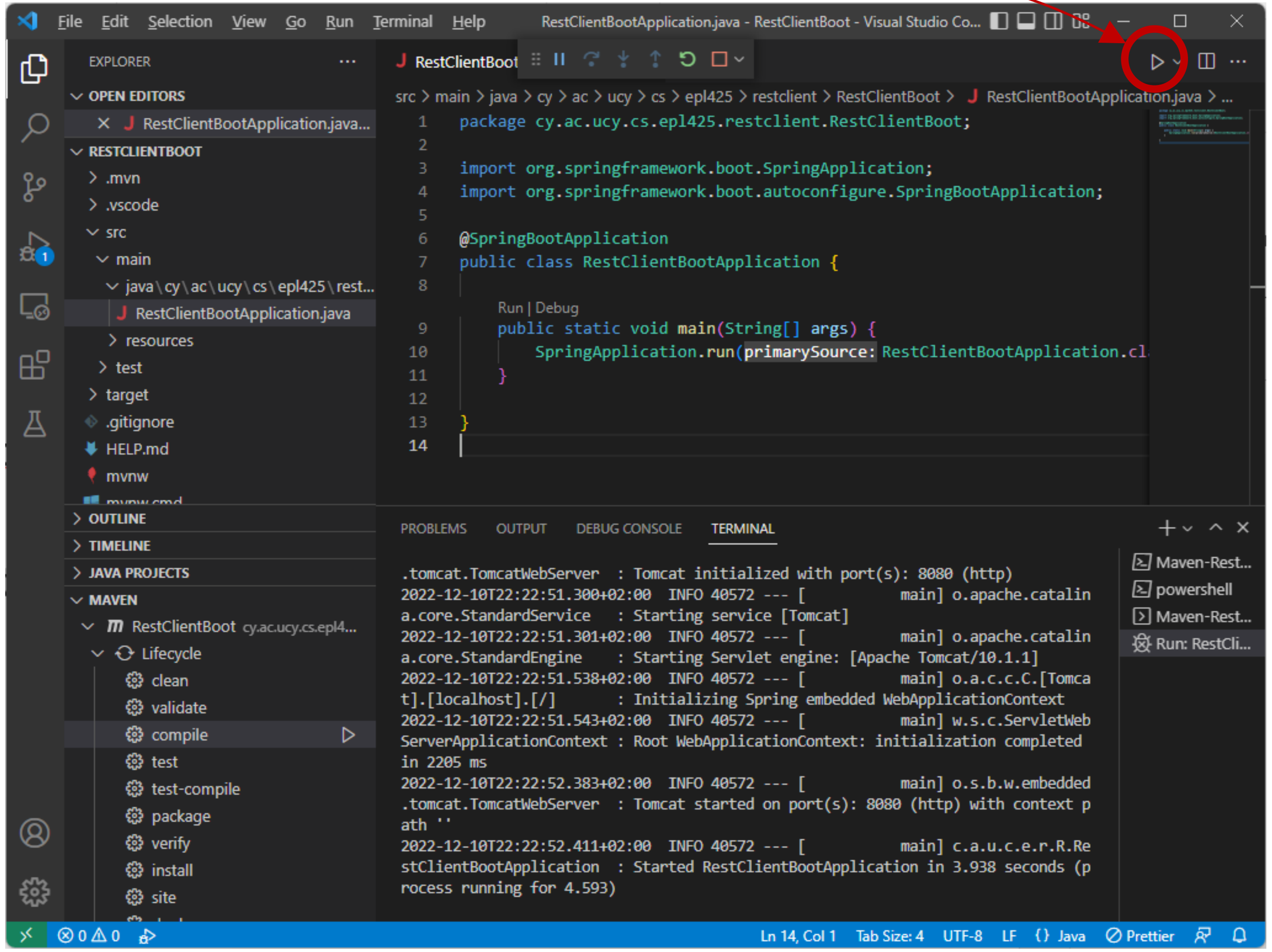

File Edit Selection View Go Run Terminal Help
RestClientBootApplication.java - RestClientBoot - Visual Studio Co...
EXPLORER
OPEN EDITORS
RestClientBootApplication.java...
RESTCLIENTBOOT
.mvn
.vscode
src
main
java\cy\ac\ucy\cs\epl425\rest...
RestClientBootApplication.java
resources
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
RestClientBoot cy.ac.ucy.cs.epl4...
Lifecycle
clean
validate
compile
test
test-compile
package
verify
install
site
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Maven-Rest...
powershell
Maven-Rest...
Ln 1, Col 2 Tab Size: 4 UTF-8 LF {} Java Prettier

```java
package cy.ac.ucy.cs.epl425.restclient.RestClientBoot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class RestClientBootApplication {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(primarySource: RestClientBootApplication.cl
    }

}
```

```
[INFO] Building RestClientBoot 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.3.0:resources (default-resources) @ RestClie
ntBoot ---
[INFO] Copying 1 resource
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.10.1:compile (default-compile) @ RestClientBo
ot ---
[INFO] Nothing to compile - all classes are up to date
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.629 s
[INFO] Finished at: 2022-12-10T22:19:44+02:00
[INFO] ------------------------------------------------------------------------
```

Run Spring Boot Maven Project

- Run the project
- The default Spring Web Boot project does not provide any functionality
- In the next few slides we will add a few Java classes to obtain data from a RESTful API


File Edit Selection View Go Run Terminal Help
RestClientBootApplication.java - RestClientBoot - Visual Studio Co...
EXPLORER
OPEN EDITORS
RestClientBootApplication.java...
RESTCLIENTBOOT
.mvn
.vscode
src
main
java \ cy \ ac \ ucy \ cs \ epl425 \ rest...
RestClientBootApplication.java
resources
test
target
.gitignore
HELP.md
mvnw
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
RestClientBoot cy.ac.ucy.cs.epl4...
Lifecycle
clean
validate
compile
test
test-compile
package
verify
install
site
src > main > java > cy > ac > ucy > cs > epl425 > restclient > RestClientBoot > RestClientBootApplication.java
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Maven-Rest...
powershell
Maven-Rest...
Run: RestCli...
Ln 14, Col 1
Tab Size: 4
UTF-8
LF
Java
Prettier

```java
package cy.ac.ucy.cs.epl425.restclient.RestClientBoot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class RestClientBootApplication {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(primarySource: RestClientBootApplication.cl
    }

}
```

```
.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2022-12-10T22:22:51.300+02:00  INFO 40572 --- [           main] o.apache.catalin
a.core.StandardService   : Starting service [Tomcat]
2022-12-10T22:22:51.301+02:00  INFO 40572 --- [           main] o.apache.catalin
a.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.1]
2022-12-10T22:22:51.538+02:00  INFO 40572 --- [           main] o.a.c.c.C.[Tomca
t].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-12-10T22:22:51.543+02:00  INFO 40572 --- [           main] w.s.c.ServletWeb
ServerApplicationContext : Root WebApplicationContext: initialization completed
in 2205 ms
2022-12-10T22:22:52.383+02:00  INFO 40572 --- [           main] o.s.b.w.embedded
.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context p
ath ''
2022-12-10T22:22:52.411+02:00  INFO 40572 --- [           main] c.a.u.c.e.r.R.Re
stClientBootApplication  : Started RestClientBootApplication in 3.938 seconds (p
rocess running for 4.593)
```

RESTful API Client in Java using Spring Boot

- Use https://nationalize.io/ RESTful API to predict the nationality of a name

```
https://api.nationalize.io?name=pavlos
```

REQUEST

```
{
      "name": "pavlos",
      "country": [
        {
         "country_id": "CY",
         "probability": 0.6239777660881337
        },
        {
         "country_id": "GR",
         "probability": 0.3572708199586962
        },
        {
         "country_id": "CZ",
         "probability": 0.00612225866824608
        }
     ]
}
```

REPLY

RESTful API Client in Java using Spring Boot

- Initalizr created class `RestClientBootApplication.java` with a main() at `src/main/java/cy/ac/ucy/cs/epl425/restclient/RestClientBoot/`
- We need to add a few other things (shaded below)

```
@SpringBootApplication
public class RestClientBootApplication {
          // A logger, to send output to the log (the console, in this example)
          private static final Logger log = LoggerFactory.getLogger(RestClientBootApplication.class);

          public static void main(String[] args) {
                    SpringApplication.run(RestClientBootApplication.class, args);
          }
          // A RestTemplate, which uses the Jackson JSON processing library to process the incoming data.
          @Bean
          public RestTemplate restTemplate(RestTemplateBuilder builder) {
                    return builder.build();
          }
          // A CommandLineRunner that runs the RestTemplate (and, consequently, fetches data) on startup.
          // Deserialize response bytes into a JAVA class: Nationalize class
          @Bean
          public CommandLineRunner run(RestTemplate restTemplate) throws Exception {
                    return args -> {
                              Nationalize nationalize = restTemplate.getForObject(
                                                  "https://api.nationalize.io?name=pavlos", Nationalize.class);
                              log.info(nationalize.toString()); // print object with REST data in logs
                    };
          }
}
```

RESTful Client Development: Spring Boot

- Create a model class e.g. Nationalize.java to accomodate the data that we will consume in

src/main/java/cy/ac/ucy/cs/epl425/restclient/RestClientBoot

```
{
        "name": "pavlos",
        "country": [
          {
          "country_id": "CY",
          "probability": 0.6239777660881337
          },
          {
          "country_id": "GR",
          "probability": 0.3572708199586962
          },
          {
          "country_id": "CZ",
          "probability": 0.0061222586682460 8
          }
        ]
}
```

Country.java (next slide)

Nationalize.java

See Here

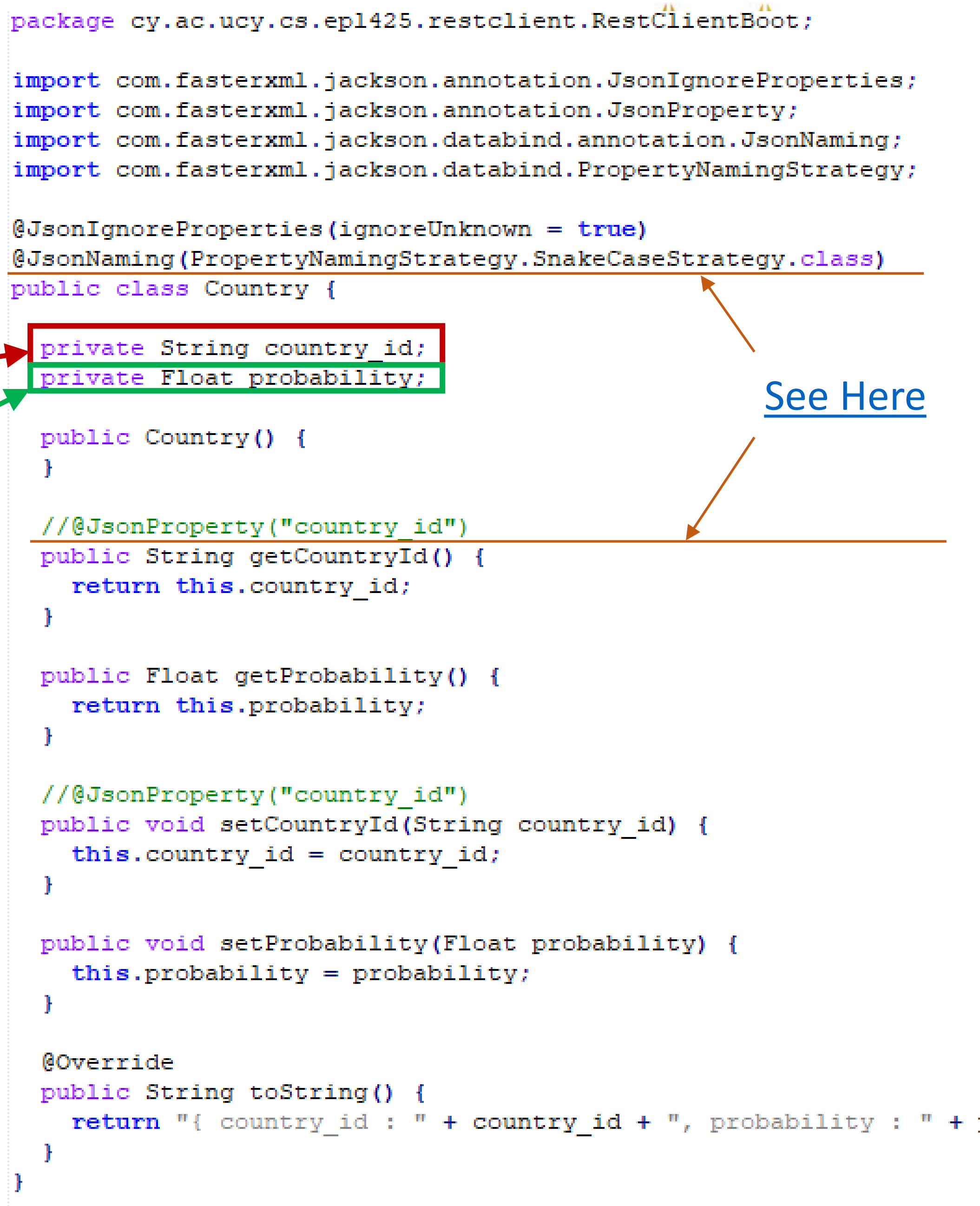
```java
package cy.ac.ucy.cs.epl425.restclient.RestClientBoot;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import java.util.List;
import java.util.ArrayList;

@JsonIgnoreProperties(ignoreUnknown = true)
public class Nationalize {

  private String name;
  private List<Country> country = new ArrayList<>();

  public Nationalize() {
  }

  public String getName() {
    return this.name;
  }

  public List<Country> getCountry() {
    return this.country;
  }

  public void setName(String name) {
    this.name = name;
  }

  public void setCountry(List<Country> country) {
    this.country = country;
  }

  @Override
  public String toString() {
    return "{" +
        "name : " + name + " " +
        ", countries : " + country + " " +
        '}';
  }
}
```

- Additional class to capture the inner country info e.g. Country.java, in the same folder:

```
src/main/java/cy/ac/ucy/cs/epl425/restclient/RestClientBoot
```

```
{
        "name": "pavlos",
        "country": [
          {
           "country_id": "CY",
           "probability": 0.6239777660881337
          },
          {
           "country_id": "GR",
           "probability": 0.3572708199586962
          },
          {
           "country_id": "CZ",
           "probability": 0.00612225866824608
          }
        ]
}
```

Country.java



```java
package cy.ac.ucy.cs.epl425.restclient.RestClientBoot;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.annotation.JsonNaming;
import com.fasterxml.jackson.databind.PropertyNamingStrategy;

@JsonIgnoreProperties(ignoreUnknown = true)
@JsonNaming(PropertyNamingStrategy.SnakeCaseStrategy.class)
public class Country {

  private String country_id;
  private Float probability;

  public Country() {
  }

  //@JsonProperty("country_id")
  public String getCountryId() {
    return this.country_id;
  }

  public Float getProbability() {
    return this.probability;
  }

  //@JsonProperty("country_id")
  public void setCountryId(String country_id) {
    this.country_id = country_id;
  }

  public void setProbability(Float probability) {
    this.probability = probability;
  }

  @Override
  public String toString() {
    return "{ country_id : " + country_id + ", probability : " + 
  }
}
```

See Here

RESTful API Client in Java using Spring Boot

- `RestClientBootApplication.java` libraries to be imported:

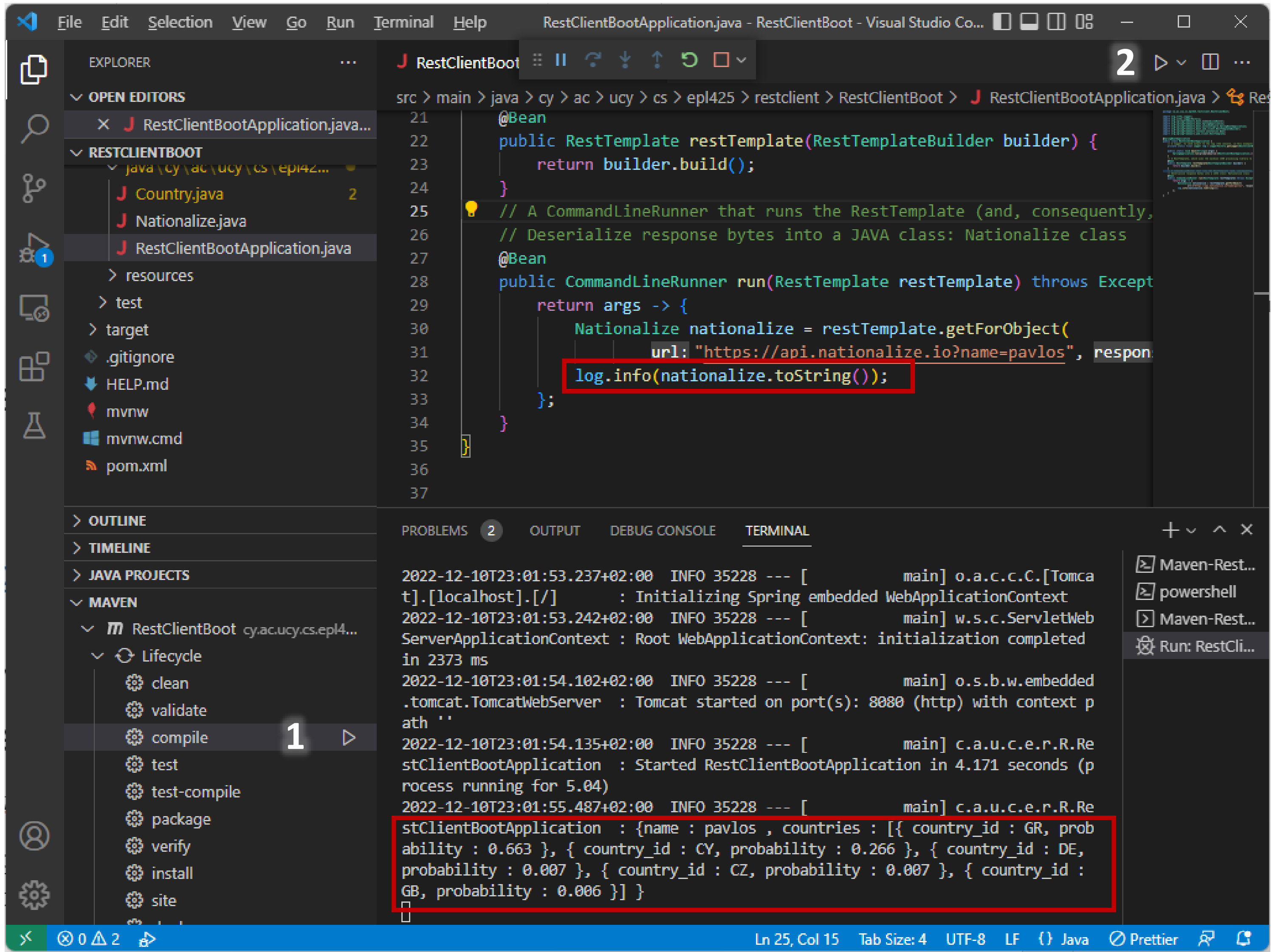
```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.client.RestTemplateBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;
```

Run Spring Boot RESTful API Client

- unzip `RestClientBoot.zip` you created through Spring Initializr
- Download from lab's website the following files:
  - Nationalize.java
  - Country.java
  - RestClientBootApplication.java

  and place them into
  `src/main/java/cy/ac/ucy/cs/epl425/restclient/RestClientBoot/`
- Compile and run application (see next slide)

Run Spring Boot RESTful API Client



```java
    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        return builder.build();
    }
    // A CommandLineRunner that runs the RestTemplate (and, consequently,
    // Deserialize response bytes into a JAVA class: Nationalize class
    @Bean
    public CommandLineRunner run(RestTemplate restTemplate) throws Except
        return args -> {
            Nationalize nationalize = restTemplate.getForObject(
                    url: "https://api.nationalize.io?name=pavlos", respon
            log.info(nationalize.toString());
        };
    }
}
```

```
2022-12-10T23:01:53.237+02:00  INFO 35228 --- [           main] o.a.c.c.C.[Tomca
t].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-12-10T23:01:53.242+02:00  INFO 35228 --- [           main] w.s.c.ServletWeb
ServerApplicationContext : Root WebApplicationContext: initialization completed
in 2373 ms
2022-12-10T23:01:54.102+02:00  INFO 35228 --- [           main] o.s.b.w.embedded
.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context p
ath ''
2022-12-10T23:01:54.135+02:00  INFO 35228 --- [           main] c.a.u.c.e.r.R.Re
stClientBootApplication  : Started RestClientBootApplication in 4.171 seconds (p
rocess running for 5.04)
2022-12-10T23:01:55.487+02:00  INFO 35228 --- [           main] c.a.u.c.e.r.R.Re
stClientBootApplication  : {name : pavlos , countries : [{ country_id : GR, prob
ability : 0.663 }, { country_id : CY, probability : 0.266 }, { country_id : DE,
probability : 0.007 }, { country_id : CZ, probability : 0.007 }, { country_id :
GB, probability : 0.006 }] }
```

APPENDIX A

JSON related

University of Cyprus
Department of Computer Science

@JsonIgnoreProperties ignoreUnknown

- When we pass `true` to `ignoreUnknown` element, then in deserialization if JSON document has a field (property) for which there is no logical property then that JSON field will be ignored, and no error will be thrown.
- Consider the following class:

```
@JsonIgnoreProperties(ignoreUnknown = true)
public class Book {
        @JsonProperty("bookId")
        private String id;

        @JsonProperty("bookName")
        private String name;

        @JsonProperty("bookCategory")
        private String category;
}  I
```

In this class we have `bookId`, `bookName` and `bookCategory` logical properties.

@JsonIgnoreProperties ignoreUnknown

- Suppose we have a JSON document with some unknown fields (properties).

```
{
  "bookId" : "A101",
  "bookName" : "Learning Java",
  "bookCategory" : "Java",
  "pubYear" : "2018",
  "price" : "200",
}
```

- In the above JSON fields, `pubYear` and `price` has no corresponding logical properties in `Book` class. In deserialization, we will not get exception because we are using `ignoreUnknown = true` in `@JsonIgnoreProperties annotation`.

@JsonNaming and @JsonProperty

- PROBLEM: Jackson (JSON library) ignores snake case JSON fields
- SOLUTIONS (any of the two):
  1. Use `@JsonNaming(PropertyNamingStrategy.SnakeCaseStrategy.class)` to define a global naming convention for JSON deserialization
  2. To directly bind your data to your custom types, you need to specify the variable name to be exactly the same as the field (property) in the JSON document returned from the API. In case your variable name and field in JSON doc do not match, you can use `@JsonProperty` annotation to specify the exact key of the JSON document

APPENDIX B

Instructions on how to Download and Install Java and Apache Maven

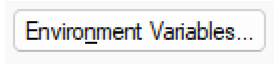
University of Cyprus
Department of Computer Science

Download and Install JDK

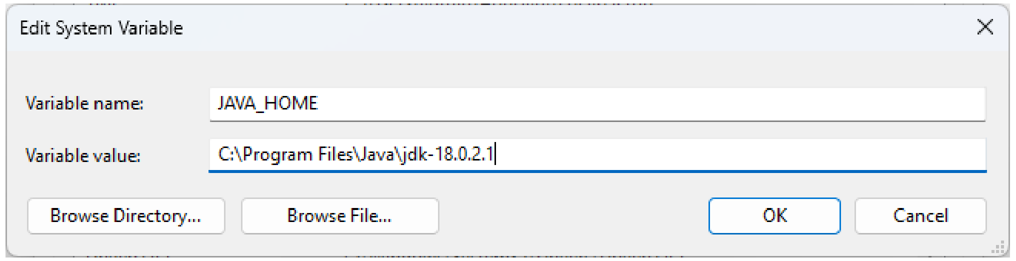
- Latest JDK installer (.exe for Windows): https://www.oracle.com/java/technologies/downloads
- Double click to install it

- For MAC users: see here for installation and for setting the environmental variable

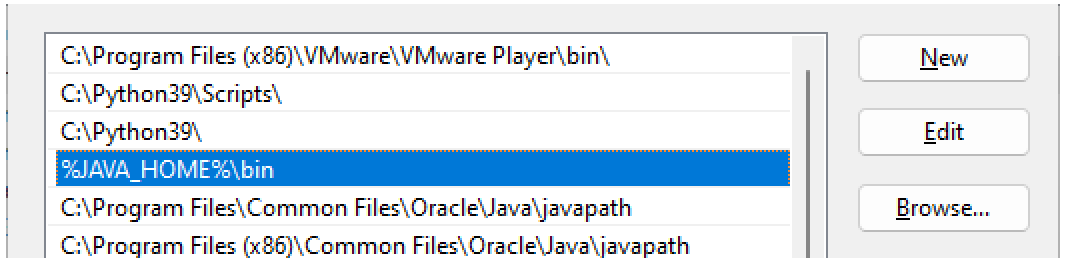
Set JAVA_HOME (for Windows)

1. Locate your JDK installation directory
   - If you didn't change the path during installation, it'll be something like C:\Program Files\Java\jdk-18.0.2.1
2. In Windows 10/11, Search for Environment Variables then select "Edit the system environment variables"
3. Click the Environment Variables button: 

4. Under System Variables, click New.
5. In the Variable Name field enter `JAVA_HOME`
6. In the Variable Value field, enter your JDK installation path (step 1)
7. Click OK
8. In System Variables, double-click on "Path"
9. Click on New and enter `%JAVA_HOME%\bin`
10. Click OK
11. Installation verification: open cmd and type `java -version` and `javac -version`



Variable name: JAVA_HOME
Variable value: C:\Program Files\Java\jdk-18.0.2.1



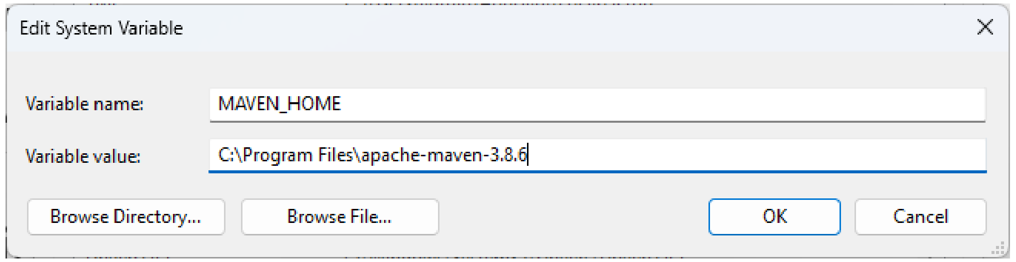
```
C:\Program Files (x86)\VMware\VMware Player\bin\
C:\Python39\Scripts\
C:\Python39\
%JAVA_HOME%\bin
C:\Program Files\Common Files\Oracle\Java\javapath
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
```

Download and Install MAVEN

- Go to https://maven.apache.org/download.cgi
- For Windows, download binary .zip archive
- For MAC, download binary .tar.gz archive
- Extract binaries and note the path
  - For Windows e.g. C:\Program Files\apache-maven-3.8.6
  - Check that the folder bin\ is within the above maven folder
- For MAC users: see here for installation and for setting the environmental variable

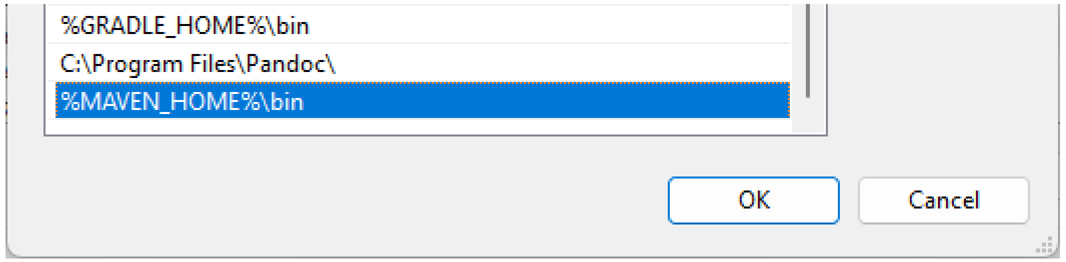
Set MAVEN_HOME (for Windows)

1. Locate your MAVEN installation directory
   - E.g. C:\Program Files\apache-maven-3.8.6
2. In Windows 10/11, Search for Environment Variables then select "Edit the system environment variables"
3. Click the Environment Variables button:
4. Under System Variables, click New.
5. In the Variable Name field enter `MAVEN_HOME`
6. In the Variable Value field, enter your MAVEN installation path (step 1)
7. Click OK
8. In System Variables, double-click on "Path"
9. Click on New and enter `%MAVEN_HOME%\bin`
10. Click OK
11. Installation verification: open cmd and type `mvn -v`



Variable name: MAVEN_HOME

Variable value: C:\Program Files\apache-maven-3.8.6



%GRADLE_HOME%\bin
C:\Program Files\Pandoc\
%MAVEN_HOME%\bin