Web and Spring Basics

Front and Back

The front-end is the parts of a webpage that a visitor can interact with and see**. The front-end** is composed of JavaScript, CSS, HTML, and other static assets, such as images or videos.

Note: You'll sometimes hear front-end development referred to as client-side development. Our instinct might be to think of the client as the human visitor or user of a website, but when referring to the client in web development, we're usually referring to the non-human requester of content. In the case of visiting a website, the client is the browser, but in other circumstances, a client might be another application, a mobile device, or even a "smart" appliance!

The back-end consists of all the behind-the-scenes processes and data that make a website function and send resources to clients.

A web server is a process running on a computer that listens for incoming requests for information over the internet and sends back responses.

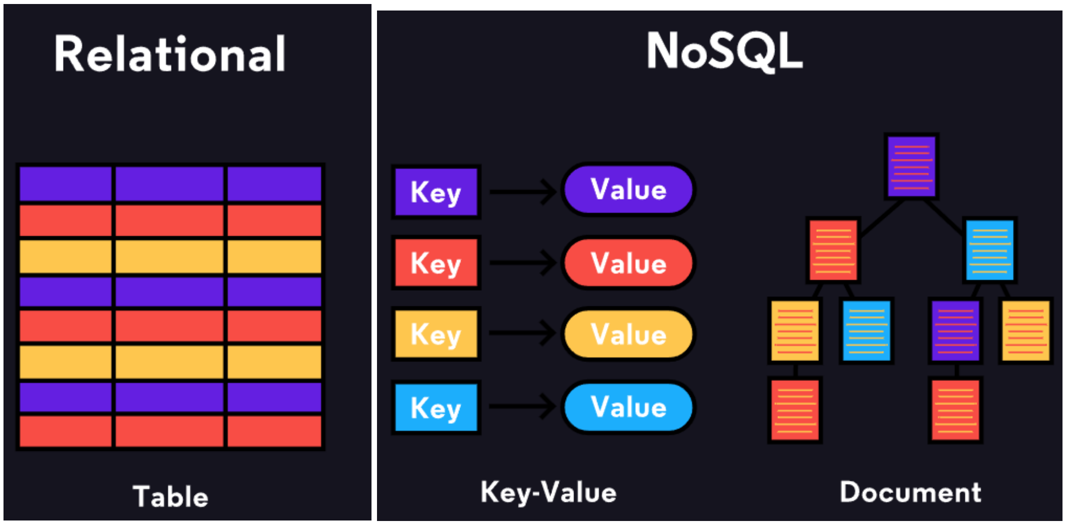
Static website is where a client makes a single request. The web server receives that request and sends the client a response containing everything needed to view the website. A website is static because once the assets are received, they don't change or move.

Note: This doesn't mean the website is not interactive.

Dynamic content is where modern web applications cater to the specific user rather than sending the same files to every visitor of a webpage

Databases are collections of information. Two types of databases:

1. **relational databases** store information in tables with columns and rows
2. **non-relational databases** might use other systems such as key-value pairs or a document storage model.



In fact, much of what the back-end entails is reading, updating, or deleting information stored in a database.

a web API is a collection of predefined ways of, or rules for, interacting with a web application's data, often through an HTTP request-response cycle.

Authentication (Who is this user? Are they who they claim to be?) is the process of validating the identity of a user. Example logins with usernames and passwords. These credentials need to be securely stored in the back-end on a database and checked upon each visit.

Authorization (Who is allowed to do and see what?) controls which users have access to which resources and actions.

Note: Developers can construct back-ends in many different languages like PHP, Java, JavaScript, Python, and more.

Framework	**Language**
Laravel	PHP
Express.js	JavaScript (runs in the Node environment)
Ruby on Rails	Ruby
Spring	Java
JSF	Java
Flask	Python
Django	Python

The collection of technologies used to create the front-end and back-end of a web application is referred to as a stack.

The MEAN stack is a technology stack for building web applications that uses MongoDB, Express.js, AngularJS, and Node.js: MongoDB is used as the database, Node.js with Express.js for the rest of the back-end, and Angular is used as a front-end framework. While the LAMP Stack, sometimes considered the archetypal stack, uses Linux, Apache, MySQL, and PHP.

HTTP Requests

HTTP stands for Hypertext Transfer Protocol and is used to structure requests and responses over the internet. HTTP requires data to be transferred from one point to another over the network. The transfer of resources happens using TCP (Transmission Control Protocol).

There are 4 basic HTTP verbs we use in requests to interact with resources in a REST system:

- **GET** — retrieve a specific resource (by id) or a collection of resources
- **POST** — create a new resource
- **PUT** — update a specific resource (by id)
- **DELETE** — remove a specific resource by id

The back-end is all of the technology required to process the incoming request and generate and send the response to the client. This typically includes three major parts:

- The server. This is the computer that receives requests.
- The app. This is the application running on the server that listens for requests, retrieves information from the database, and sends a response.
- The database. Databases are used to organize and persist data.

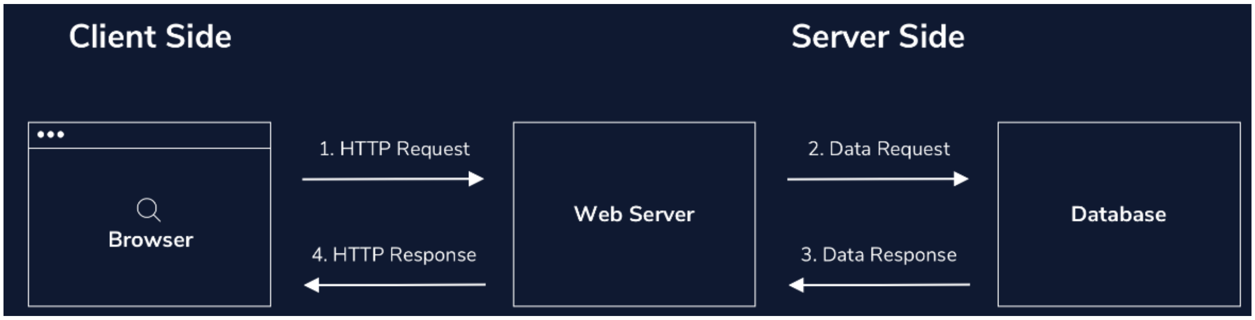
A server is simply a computer that listens for incoming requests.

Spring

Spring is an open-source Java framework that is useful, among other things, for building RESTful web applications.

Spring contains templates for many different kinds of applications (including Spring Cloud, Spring Web Services, Spring Security, etc.).

How spring work HTTP get request example

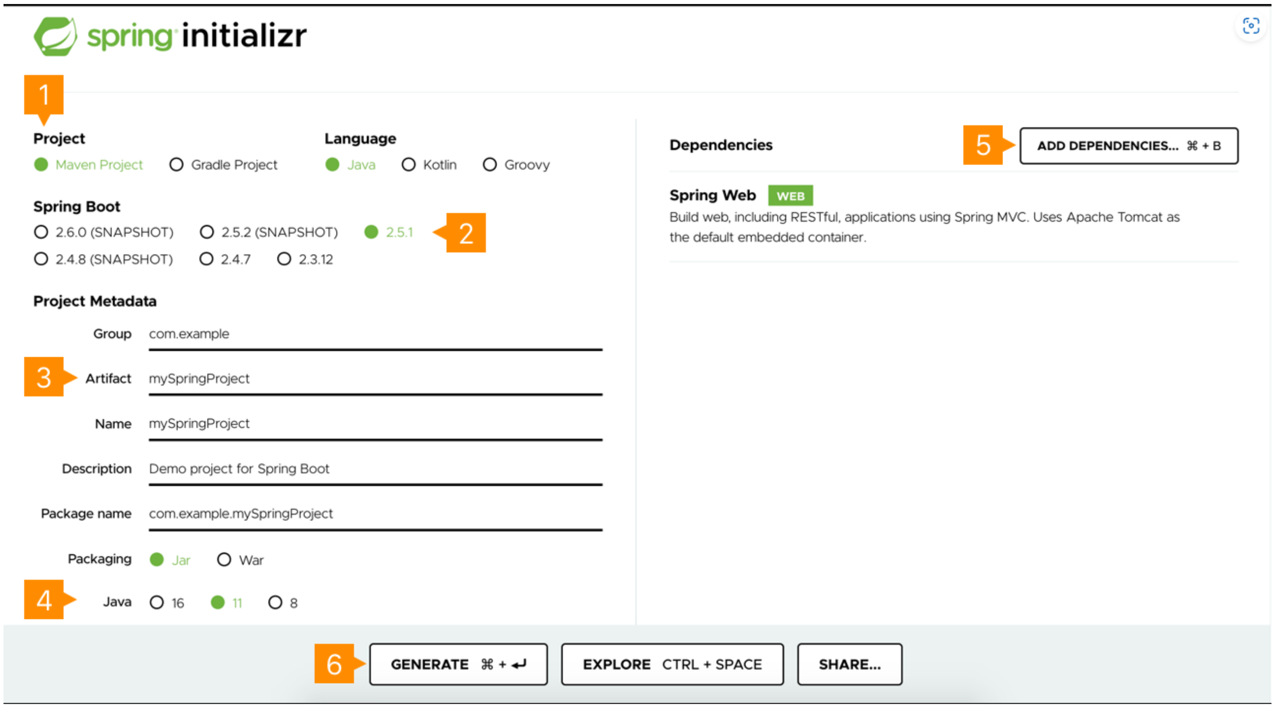


Create Spring Boot web applications

Spring Boot is one of the easiest ways to build Spring web projects, enabling applications to run with minimal configuration.

Spring Boot is an embedded Spring framework that allows web applications to run with minimal configuration and code.

navigate to **start.spring.io** to start a new Spring Boot web project.



Project
● Maven Project ○ Gradle Project

Language
● Java ○ Kotlin ○ Groovy

Spring Boot
○ 2.6.0 (SNAPSHOT) ○ 2.5.2 (SNAPSHOT) ● 2.5.1
○ 2.4.8 (SNAPSHOT) ○ 2.4.7 ○ 2.3.12

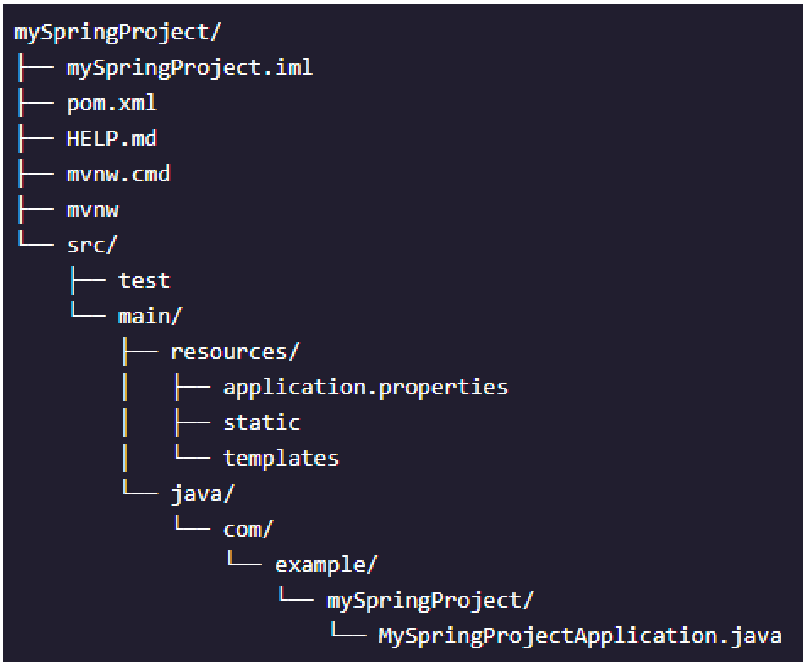
Project Metadata
Group: com.example
Artifact: mySpringProject
Name: mySpringProject
Description: Demo project for Spring Boot
Package name: com.example.mySpringProject
Packaging: ● Jar ○ War
Java: ○ 16 ● 11 ○ 8

Dependencies ADD DEPENDENCIES... ⌘ + B

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

GENERATE ⌘ + ↵ EXPLORE CTRL + SPACE SHARE...

Maven folder structure



```
mySpringProject/
├── mySpringProject.iml
├── pom.xml
├── HELP.md
├── mvnw.cmd
├── mvnw
└── src/
    ├── test
    └── main/
        ├── resources/
        |   ├── application.properties
        |   ├── static
        |   └── templates
        └── java/
            └── com/
                └── example/
                    └── mySpringProject/
                        └── MySpringProjectApplication.java
```

pom.xml

Directly under the main folder “mySpringProject” is a file named pom.xml. POM stands for Project Object Model – an XML file that specifies and loads important project data, such as configuration, build profiles, and dependencies. All Maven projects are built using its own POM.

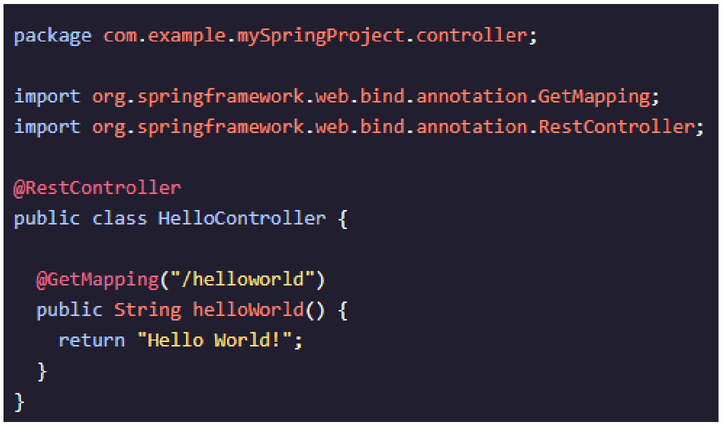
application.properties

application.properties is a configuration file that specifies the properties of your Spring application. Examples of application properties can include JSON properties, data properties, server properties, and more. A Spring application will read and load properties specified in application.properties during project build.

Implement Spring Code

Under the folder src/main/java/com/example/mySpringProject make a separate controller folder and make a new file HelloController.java.

Copy-and-paste the code below into HelloController.java.



```java
package com.example.mySpringProject.controller;

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

@RestController
public class HelloController {

  @GetMapping("/helloworld")
  public String helloWorld() {
    return "Hello World!";
  }
}
```

In the code snippet above, @RestController and @GetMapping are examples of different kinds of annotations. Annotations, recognized by their starting @ symbol, are built-in units of Spring code that make complex functionality readily available in Spring applications. For example, the @GetMapping("/helloworld") annotation means that the annotated method will be executed every time the app receives a GET request to the /helloworld endpoint.

Testing with the Browser

Use the url http://localhost:8080/helloworld

Testing with Curl

terminal window, type curl http://localhost:8080/helloworld

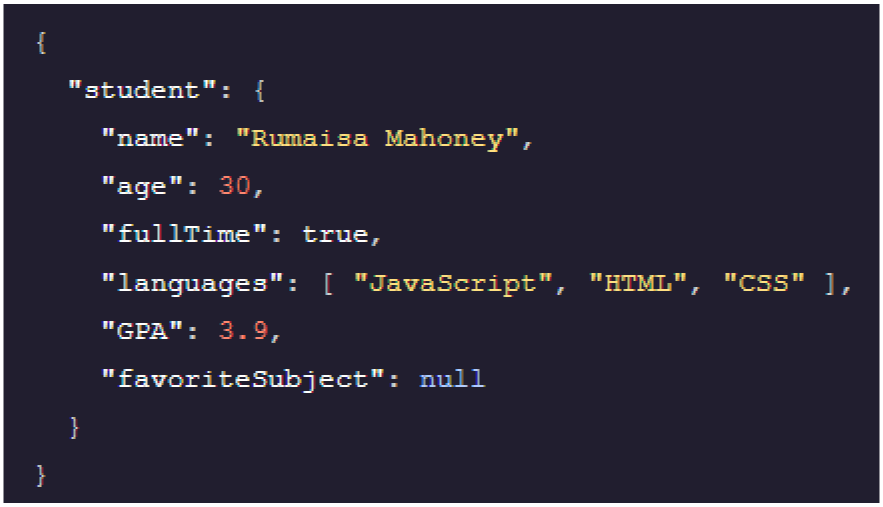
./mvnw spring-boot:run uses Maven to run the Spring Boot project, which is a common way to launch Spring Boot applications. It specifically executes the Spring Boot plugin goal within Maven to start the project.

JSON

JSON, or JavaScript Object Notation, is a popular, language-independent, standard format for storing and exchanging data.

JSON is heavily used to facilitate data transfer in web applications between a client, such as a web browser, and a server.

JSON Syntax: A sample JSON object is represented as follows:



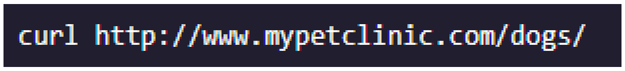
```
{
  "student": {
    "name": "Rumaisa Mahoney",
    "age": 30,
    "fullTime": true,
    "languages": [ "JavaScript", "HTML", "CSS" ],
    "GPA": 3.9,
    "favoriteSubject": null
  }
}
```

What is curl?

curl is a command-line tool for transferring data from or to a server. It supports multiple protocols, including HTTP.

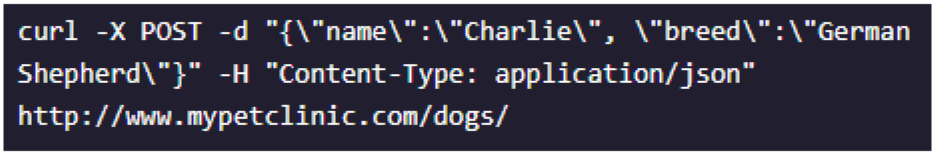
curl provides an easy way to make requests to a server from the command line or from inside a script. It is especially helpful for testing APIs while building applications because making a request with curl allows you to hit endpoints for which the corresponding user interfaces do not yet exist.

Curl is useful for testing HTTP request methods from your local command line.



```
curl http://www.mypetclinic.com/dogs/
```

Post curl

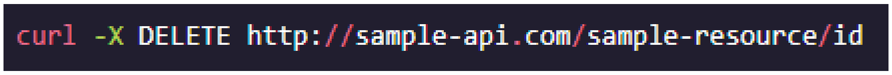


```
curl -X POST -d "{\"name\":\"Charlie\", \"breed\":\"German
Shepherd\"}" -H "Content-Type: application/json"
http://www.mypetclinic.com/dogs/
```

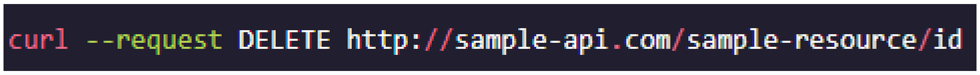
1. -X POST tells the server that the client is making a POST request, where -X is the curl parameter specifying the type of request method to use.
2. -d (short for --data) indicates to the server that the client is sending new data to an existing application.
3. The content inside "{ }", \"name\":\"Charlie\", \"breed\":\"German Shepherd\" , is the data that the client wants to send (as indicated by the preceding -d). We use the {\"key1\":\"value1\", \"key2\":\"value2\"} syntax, where name and breed are the names of the two keys that define the dog object, and “Charlie” and “German Shepherd” are the values corresponding to those keys respectively.
4. -H "Content-Type: application/json" specifies that we are sending data in JSON format.

5. Finally, the URL http://www.mypetclinic.com/dogs/ tells the server where to send this new data. In this case, the newly-created dog object will be posted under the dogs resource.

By default, curl makes GET requests, but you can use the -X option (short for --request) followed by the HTTP verb (eg. PUT, POST, DELETE, etc.) of your choice to make other types of requests.

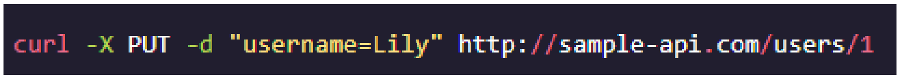


```
curl -X DELETE http://sample-api.com/sample-resource/id
```



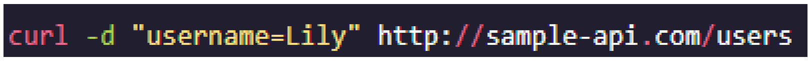
```
curl --request DELETE http://sample-api.com/sample-resource/id
```

To send data in the body of a request made with curl, you can use the -d option (short for --data) followed by the data you wish to send. For example, to update the name of the user with id=1, run:



```
curl -X PUT -d "username=Lily" http://sample-api.com/users/1
```

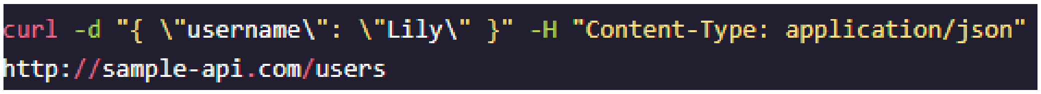
Note that if you include the -d option and do not specify an HTTP verb, curl will default to sending a POST request.



```
curl -d "username=Lily" http://sample-api.com/users
```

By default, curl sends POST/PUT requests with the content-type application/x-www-form-urlencoded. If you are using curl to make a request to an API that expects to receive data in JSON format, you must set the request's content-type to application/json by adding a request header.

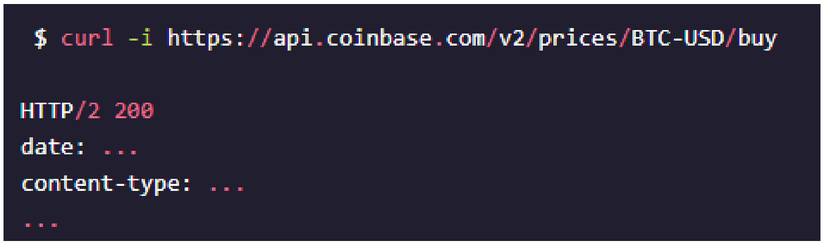
Use the -H (short for --header) option, followed by the header you wish to add to your request,



```
curl -d "{ \"username\": \"Lily\" }" -H "Content-Type: application/json"
http://sample-api.com/users
```

The above curl command essentially makes the same request as the previous one but sends the data as JSON instead of a key=value pair.

Adding the -i option (short for --include) to your curl commands will cause curl to print the response header to the command shell in addition to the response body.



```
 $ curl -i https://api.coinbase.com/v2/prices/BTC-USD/buy

HTTP/2 200
date: ...
content-type: ...
...
```

HTTP response status codes

1. Informational responses (100 – 199)
2. Successful responses (200 – 299)
3. Redirection messages (300 – 399)
4. Client error responses (400 – 499)
5. Server error responses (500 – 599)