API

Monday, April 4, 2022 2:14 PM

- API helps us to send the data to the server and using the data that we provide , some data is generated and send to the user and accordingly the user page is changed.\
- Here all the business logics and all the instructions and responses from and through user interface and database will be managed by the API.

The primary or most commonly-used HTTP methods are **POST, GET, PUT, PATCH, and DELETE**. These methods correspond to create, read, update, and delete (or CRUD) operations, respectively.

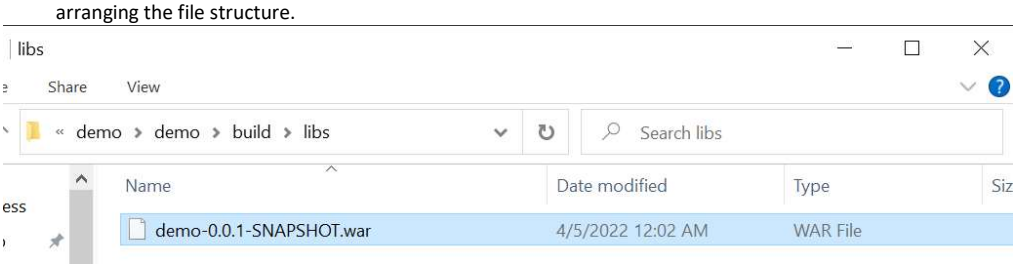
Build Framework: Maven , Gradle , Ant

- Frame work which is used to develop , manage, and pack them to develop and deploy on any webserver.
- For resolving dependencies , for example if we want to establish the connection of the database we no need to write the basic logic from the scratch , these build frameworks provide a template for us to utilize these functions, without typing the code . Using this size of the code will also be decreased.
- These provides us with re usable components.
- Maven is basically xml file development.
- Build framework is used for downloading and managing dependencies and how packaging must be done.
- Docker is used for deployment. Build framework is used for package management.

What is Backend?

Backend. In the computer world, the "backend" refers to any part of a website or software program that users do not see. It contrasts with the frontend, which refers to a program's or website's user interface. In programming terminology, the backend is the "data access layer," while the frontend is the "presentation layer.".
Using spring boot we can develop the application very fast. And we can create API very fast.

- To establish a network between database and user interface we use API to get information from and through.
- Rest API is stateless , it mean the input can be anything ppt, json , excel or any format. So Rest API is much preferable now a days.
- Maven dependencies declared in pom file, maven use xml script. But people shifts from xml to the groovy . Groovy language is used in Gradle .
- For Gradle we will have groovy script to declare the dependencies and required jars which are need to run the application on the server.
- We can have both Gradle and maven in the same project simultaneously.
- Main benefit of this is , we no need to concentrate on downloading dependencies and by arranging the file structure.



If we execute the Gradle project by build war under the build section then our Gradle project gets executed and the following war file gets designed. And we can push the following war file to tomcat server to run the application on the server. We need to create a controller class so that we can utilize the URL's.

- All the classes that are interacting with the API will be given in controller section which is user defined.
- We need to run build the project first and then boot war the project and then we can deploy the war file on the tomcat server.

Steps to work on simple API:

1) Spring Initializr **use this link to download Gradle file with spring web dependency**
2) Download the file and extract it it , the file will be named as demo zip file . So extract that file.
3) Do open the project using eclipse, up ecto following path.

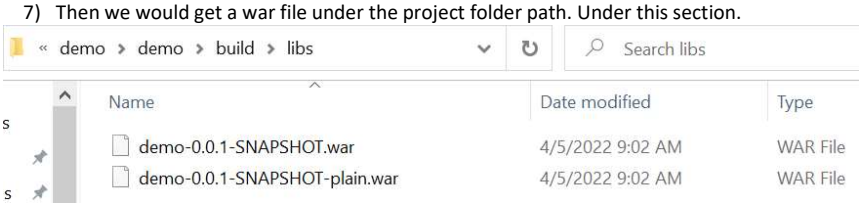
4) You must have such structure of folder , Mainly check that you have Gradle on your folder and create DemoApplication.java and servletinitializer.java and demo controller.java . Demo

Application and Servlet Initializer will be automatically in there. So we need to create demo controller.java file under controller package.

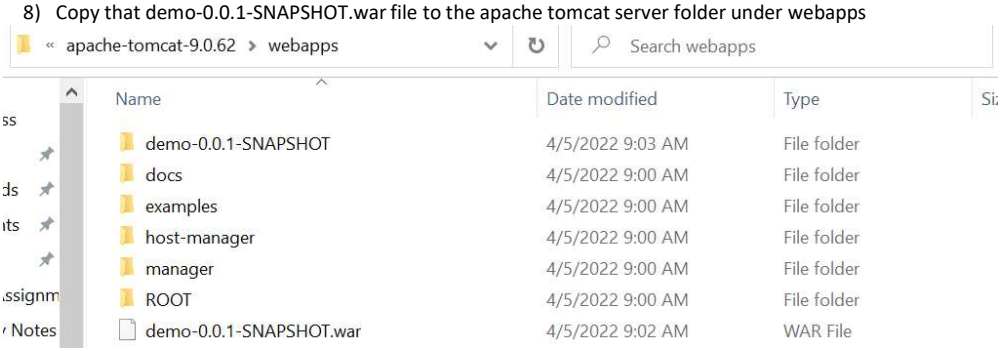
5) Have the following code under DemoController.java

```
package com.example.demo.controller;



import java.util.Date;



import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;



/* import com.example.demo.model.User; */



@RestController
@RequestMapping(value = "/demo")
// @ConditionalOnProperty(prefix = "application", name = "executeIntegrationTest",
// havingValue = "true")
public class DemoController {

	private final Logger LOG = LoggerFactory.getLogger(getClass());

	@RequestMapping(value = "/sayHello", method = RequestMethod.GET)
	public String/*ResponseEntity<Object>*/ addNewUsers() {
		LOG.info("received request" + new Date());
		return "Helllooo";
	}
}
```

6) And now we need to run the build and boot war under build section of project under Gradle tasks.
7) Then we would get a war file under the project folder path. Under this section.



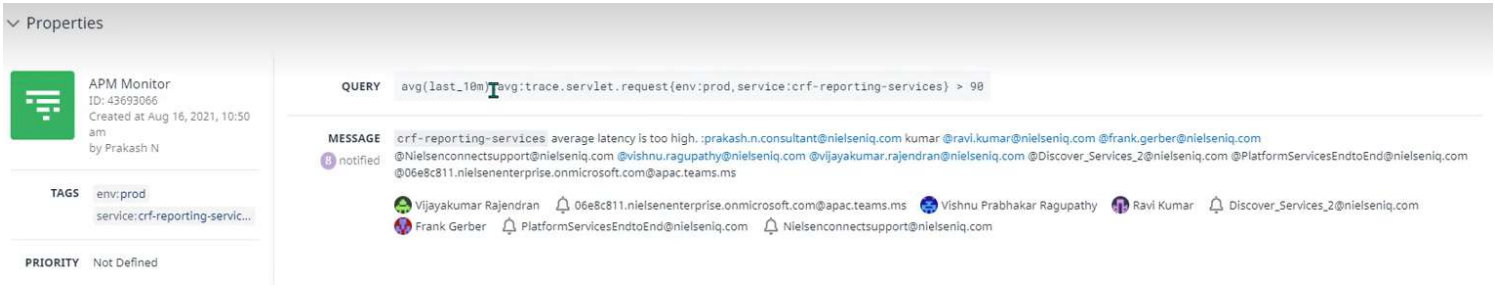
8) Copy that demo-0.0.1-SNAPSHOT.war file to the apache tomcat server folder under webapps



9) To run the tomcat server , run cmd under following path and execute startup.bat then tomcat starts execute and if we want to stop it we need to click shutdown command under cmd.
10) Check localhost:8080 when we start the tomcat server , to check whether the server initiated or not.
11) Now to check our output go to localhost:8080/demo-0.0.1-SNAPSHOT/demo/sayHello
12) Then we get the output on the browser. We initiated the output using API concept.

Helllooo

- Data Dog is used to monitor the log files and helps us in resolving errors using the log files. Datadog provide the details such as CPU consumption ,Ram allocation, latency etc.
- We can monitor alerts in data do



For example in last 10mins if we are having latency of 90 seconds to process a request we need to mail the following email id's . It means we are declaring the events which are mapped to following instructions to do. That will send an alert to raise issue in real time.
Even we can trace number of input requests for API through data dog.