@PropertySource("classpath:/user.properties")

* Xml Base Configuration Approach
* Annotation Base Approach
* Java Configuration Approach

@SpringBootApplication

@EnableAutoConfiguration

@ComponentScan

@Configuration

XML based

Annotation based <context:config>

Java source based

ORM:Object relational mapping is the method of querying and manipulating

    from a database using object oriented programming langage.

2006 JPA(java persitence API)

    Hibernate implements a standard of JPA specification

    JPA is java specification that provides an object mapping   facility to relational database for java application.

    it uses the ORM--Hibernate toplink etc....

    JPA specification for persisting, managing ,reading data from    java  object to tables in the database.

    JPA represent how to define POJO-plain old java Object as an    entity

JDBC Properties:

    javax.persistence.jdbc.driver-jdbc:mysql://localhost:3306/mydb

    javax.persistence.jdbc.URL

    javax.persistence.jdbc.user

    javax.persistence.jdbc.password

    JPA--For managing relational                                    objects in the database

    it is defined using javax.persistence

    JPA is the JAVA specification and not the implementaion

    Hibernate--Is an ORM                                        database in java applicationtool used for saving the state of the java

    it is defined using org.hibernate package

    Hibernate is an implemetion of JPA

The  
Java Persistence API ( is one approach to ORM Via JPA the  
developer can map, store, update and retrieve data from relational databases  
to Java Objects and vice versa, JPA permits the developer to work directly  
with objects rather then with SQL statements JPA is a specification and  
several implementations are available

The  
strategy attribute must be a value from the GeneratorType enumeration,  
which defines four types of strategy constants.  
1.  
AUTO : (Default) JPA decides which generator type to use, based on the  
database’s support for primary key generation  
2.  
IDENTITY The database is responsible for determining and assigning the  
next primary key  
3.  
SEQUENCE Some databases support a SEQUENCE column type  
4.  
TABLE : This type keeps a separate table with the primary key values.

<persistence  
unit> : It has the name attribute specifies a name that can be  
referenced from your Java code. The transaction type attribute informs ORM about  
transaction management. It may take values like:  
a.  
RESOURCE\_LOCAL : Application will handle transaction management. i.e.  
creating, starting and closing of transactions.  
b.  
JTA : JEE server Container will take care for transaction management.

@Component is an annotation that allows Spring to detect our custom beans automatically. In other words, without having to write any explicit code, Spring will: Scan our application for classes annotated with @Component. Instantiate them and inject any specified dependencies into them.

Spring @Bean Annotation is applied on a method to specify that it returns a bean to be managed by Spring context

@Configuration : This annotation indicates that the class has @Bean definition methods.

One of the most important annotations in spring is @Configuration annotation which indicates that the class has @Bean definition methods. So Spring container can process the class and generate Spring Beans to be used in the application. This annotation is part of the spring core framework.

@ControllerAdvice is a specialized form of the spring's stereotype annotation which allows handling exceptions across the whole application in one global handling component. Think of this as an interceptor of exceptions thrown by methods annotated with @RequestMapping.

The @ExceptionHandler is an annotation used to handle the specific exceptions and sending the custom responses to the client. Define a class that extends the RuntimeException class. You can define the @ExceptionHandler method to handle the exceptions as shown.

It is used to mark the class as a service provider. So overall @Service annotation is used with classes that provide some business functionalities. Spring context will autodetect these classes when annotation-based configuration and classpath scanning is used. Procedure. Create a Simple Spring Boot Project

Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection. Autowiring can't be used to inject primitive and string values. It works with reference only

@Autowired is a Spring annotation used to automatically inject dependencies into a Spring-managed bean.

The Dependency Injection is a design pattern that removes the dependency of the programs. In such case we provide the information from the external source such as XML file. It makes our code loosely coupled and easier for testing. In such case we write the code as:

Where is application properties in spring boot?  
Spring Boot Properties - javatpoint  
Spring Boot Framework comes with a built-in mechanism for application configuration using a file called application. properties. It is located inside the src/main/resources folder, as shown in the following figure. Spring Boot provides various properties that can be configured in the application.

The @RequestBody annotation indicates that Spring should deserialize a request body into an object. This object is passed as a parameter in the handler method. To understand the working let's create an Employee domain class and corresponding controlelr handler

DTO ( Data Transfer Object) Overview  
Data Transfer Object Design Pattern is a frequently used design pattern. It is basically used to pass data with multiple attributes in one shot from client to server, to avoid multiple calls to a remote server.

When a method annotated with @Transactional is called, a transaction will be started before the method is executed, and any changes made to the database within the method will be persisted to the database when the transaction is committed. If an exception is thrown within the method, the transaction will be rolled back, and any changes made to the database within the method will be discarded.

The MockitoAnnotations.openMocks(this) method is used in JUnit 4 to initialize the annotated fields annotated with Mockito annotations, such as @Mock, @InjectMocks, and @Spy. It is part of the Mockito framework and is used to simplify the initialization of mocks and spies in test classes.

@Mock: This annotation is used to create a mock object of a class or interface. When a field is annotated with @Mock, Mockito creates a mock instance of the corresponding class or interface.

@InjectMocks: This annotation is used to inject the mocked dependencies into the system under test. When a field is annotated with @InjectMocks, Mockito attempts to inject the mock objects (annotated with @Mock) into the corresponding fields of the class being tested.

In Angular, tsconfig.json is a configuration file that specifies how the TypeScript compiler should compile TypeScript code into JavaScript.

In Angular, angular.json is a configuration file that specifies the project configuration and build settings for an Angular application.

In an Angular app, package. json is a file that contains information about the project and its dependencies.   
Package. JSON holds all of the “npm” packages installed for the project. Angular. JSON holds the configuration for the project

@NgModule is a decorator in Angular that is used to define a module. In Angular, a module is a container for a group of related components, directives, pipes, and services.

n Angular, a guard is a class that implements the CanActivate interface and is used to control access to a specific route or URL.

Routing in Angular allows the users to create a single-page application with multiple views and allows navigation between them

Session storage is a web storage API that allows you to store key-value pairs in the user's browser for the duration of a session.

Local storage is a web storage API that allows you to store key-value pairs in the user's browser for an indefinite period of time.

In Angular, an Observable is a powerful data-handling technique that allows you to manage asynchronous data streams.

In Angular, subscribe is a method that is used to consume values emitted by an Observable.

The HttpClient module is an Angular module that provides a way to make HTTP requests to a server or external API.

Spring Security is a powerful and highly customizable security framework for Java applications

Here are the lifecycle hook methods in Angular:

ngOnChanges(): This method is called whenever one or more of the component's input properties change. It receives a SimpleChanges object that contains information about the changes.

ngOnInit(): This method is called once when the component is first initialized. It is a good place to perform any initialization logic, such as fetching data from a server or setting default values.

ngDoCheck(): This method is called during every change detection cycle, which happens frequently in Angular. It allows you to perform custom change detection logic and optimize your component's performance.

ngAfterContentInit(): This method is called after the component's content has been projected into its view. It is a good place to perform any setup that requires access to the component's child content, such as initializing child components.

ngAfterContentChecked(): This method is called after every change detection cycle that involves the component's content. It allows you to perform custom change detection logic on the component's content.

ngAfterViewInit(): This method is called after the component's view has been initialized. It is a good place to perform any setup that requires access to the component's DOM, such as setting up event listeners.

ngAfterViewChecked(): This method is called after every change detection cycle that involves the component's view. It allows you to perform custom change detection logic on the component's view.

ngOnDestroy(): This method is called just before the component is destroyed. It is a good place to perform any cleanup logic, such as unsubscribing from event listeners or cancelling pending requests.

Template-driven forms are based on HTML templates and use Angular directives to create the form structure and validation rules.

Reactive forms are based on TypeScript code and use reactive programming constructs to create the form structure and validation rules.

JPA (Java Persistence API) is a Java specification for object-relational mapping (ORM) that allows Java developers to map Java objects to relational database tables. It provides a standard way to work with relational databases in Java applications and is a part of the Java EE (Enterprise Edition) platform.

ORM stands for Object-Relational Mapping. It is a programming technique that allows you to map data between object-oriented programming languages and relational database systems. ORM frameworks provide a way to work with databases using object-oriented programming concepts, such as classes, objects, and inheritance, rather than requiring developers to write SQL queries directly.

Hibernate is an open-source Object-Relational Mapping (ORM) framework for Java. It is one of the most widely used ORM frameworks and provides a high-level, object-oriented API for working with relational databases. Hibernate is built on top of the Java Persistence API (JPA) and provides additional features and functionality beyond the JPA specification.

as akash was saying this application has various features   
here is an our application er diagram which shows relationship between the entities  
The main entities in the ER diagram include books, authors, publishers, customer

The Books entity represents the books that are available in the store, and includes attributes such as BookID, Title, AuthorID, PublisherID, Price, and Stock. The Authors entity represents the authors of the books, and includes attributes such as AuthorID, FirstName, and LastName. The Publishers entity represents the publishers of the books, and includes attributes such as PublisherID and Name.

The Customers entity represents the customers who visit the book store, and includes attributes such as CustomerID, FirstName, LastName, Email, and Phone. The Orders entity represents the orders placed by the customers, and includes attributes such as OrderID, CustomerID, OrderDate, TotalAmount, and Status. The Employees entity represents the employees who work in the book store, and includes attributes such as EmployeeID, FirstName, LastName, Email, and Role.

The relationships between these entities are defined by the foreign keys that link the tables together. For example, the Books table has a foreign key called AuthorID that links it to the Authors table, and a foreign key called PublisherID that links it to the Publishers table.

look at cadinality For example, the cardinality of the relationship between the Orders and Customers entities is one-to-many, meaning that one customer can place many orders

**package.json: it contains all the packages that are required to run this angular project**

**angular.json: Angular.json is a configuration file in an Angular project that defines various settings and options for building, testing, and deploying the application.**

**Router-outlet:**

**In Angular, router-outlet is a directive that acts as a placeholder for dynamically loaded components based on the current URL route. It is a part of the Angular Router module, which is used to manage navigation and routing in an Angular application.**

**app-root:**

**In Angular, app-root is the selector for the root component of an Angular application. It is defined in the index.html file, which is the main entry point for an Angular application.**

**decorator:**

**what is data binding:**

/\* Data Binding:

            Helps us to communicate b/w template and component

               I) one way

                  1.transfers data from component to template

                    string Interpolation

                    property                       // instead string interpolation we can use ["innerText"]

                   2.transfers data from template to component

                     event binding

               II)Two way binding

                  [(NgModel)]

Directives:

    Directives are instructions in the DOM

    a directive can be created with @Directive decorator

    3 built-in directives

//structural directives

    1)\*ngIf

         If and else

               <p \*ngIf="serverCreated; else noServer>server was created</p>

               <ng-template #noServer>

               <p>no server created</p>

               </ng-template>

    2)\*ngFor

    3)ng-switch

//Attribute directives

    4)[ngClass]="{odd:odd%2 !==0}"

    5)[ngStyle]="{backgroundColor:getColor()}"

@input()-> transfers data from child to parent

@input() servercreated:string;

//create customized event

   HTML:

   (servercreated)=create();

   TS:

   servercreated=new EventEmitter<{{serverName:string}}>();

Spring @Bean Annotation is applied on a method to specify that it returns a bean to be managed by Spring context

**The @SpringBootApplication Annotation**

The @SpringBootApplication annotation is a meta-annotation that combines three essential annotations into one:

1. @Configuration: This annotation indicates that the class has @Bean definition methods. These methods create and configure beans managed by the Spring container. With this annotation, you can define beans in the same class where you define your main application class.
2. @ComponentScan: This annotation tells Spring where to scan for components. By default, it scans the package where the main application class is located and its sub-packages, finding and registering Spring beans.
3. @EnableAutoConfiguration: This annotation enables Spring Boot's automatic configuration mechanism. It automatically configures various beans and settings based on the project's classpath, dependencies, and properties.

# Microservices:

Microservices:

Microservices is an approach to developing a single application as a suite of small services, each running on its own process and communicating with light weight mechanisms, built around business capabilities and independently deployable by fully automated deployment machinery.

Rest clients: -

Rest Template

Web Client

Feign Client (open feign)

Load balancer: -

Ribbon – developed by Netflix.

Spring cloud load balancer

Service discovery & registration: -

Eureka

Zookeeper client side service discovery & registration.



Consul

Nginx Server side service discovery & registration.



API Gateway: -  
  
Security using Key Cloak: -

Spring cloud Circuit breaker:-

Fail fast

Closed half open open

Supported Implementations: -

Netflix Hystrix.

Sentinel.

Spring Retry.

Resilience4J-> Resilience4J is a lightweight, easy to use fault tolerance library inspired by Netflix Hystrix.

@CircuitBreaker @TimeLimiter @Retry and fallback mechanisms.

Distributed Tracing: -

It helps us to trace the request from start from finish. If request is failed at any point ,we can find why it is failed and where it is failed.

traceId 🡪

Spring cloud Sleuth: -

Spring cloud Sleuth provides Spring boot auto-configuration for distributed tracing.

Generates Span Id and Trace Id.

API Gateway-----------------🡪 Order Service ----------------🡪 Inv. Service

------Span1-------- ----Span2-------- ---------span3-----------

Trace-> TraceId.

Zipkin:- Zipkin is a distributed tracing system, it helps gather timing data needed to troubleshoot latency problems in service architectures.

is a tool to visualize this information.

9411 – port no.

Event driven architecture: -

Kafka:- producer consumer

Monitoring microservices: -

Graphite, InfluxDB, openTSDB ----------------🡪 are like few Prometheus

Prometheus: - as a data source

Grafna: -

Spring boot app --------------------🡪 Prometheus ----------------------------------------------🡪Grafna

Polls every x sec polls every X sec

For metrics for metrics

**Kafka: -**

Apache kafka is an open source distributed event streaming platform.

Creating real time stream

Processing real time stream

Kafka

RabbitMQ

Raddis

**Kafka components: -**

Producer --🡪 source of data,will responsible for publishing data

Consumer --🡪act as a receiver, responsible for consuming messages

Broker 🡪 kafka server or broker, it just an intermediate entity that helps in message exchanges between a producer and consumer.

Cluster 🡪 group of servers.

Topic 🡪multiple topics to store different kind of messages, specifies the category of message or the classification of message.

Partitions 🡪 partition of topic

Offset 🡪in Kafka, a sequence number is assigned to each message in each partition of a kafka topic. This sequence number is called topic.

Consumer groups 🡪group of consumers, multiple consumers combined to share workload

Zookeeper 🡪 zookeeper is a prerequisite for kafka, kafka is a distributed system and it uses zookeeper for coordination and to track the status of kafka cluster nodes. It also keeps track of kafka topic, partitions, offsets etc.

Steps: -

Start zookeeper – 2181 port

Start kafka -- 9092 port

Create topic

KRaft Mode : - Apache kafka no longer requires zookeeper.

You can achieve various benefits like

1. Eliminating system complexities
2. Data redundancy while running kafka without zookeeper
3. Simplified Kafka architecture without any third party service dependencies.