Firestore

1. What is firestore?
2. Difference between firebase & firestore?
3. Example of firestore?

Firestore: firestore is a **NOSQL document DB** provided by **Google cloud platform**. It is designed to **store, sync, and querying data** for **web, mobile and server applications**. Firestore is a part of **firebase**, which is a **comprehensive app development platform** offered by **Google**.

Firestore uses a flexible and scalable data model based on **documents** and **collections**. Data stored in **documents**, which are **essentially key-value pairs**, and **documents** are **organized into collections**. Each document has a **unique identifier within its collection**.

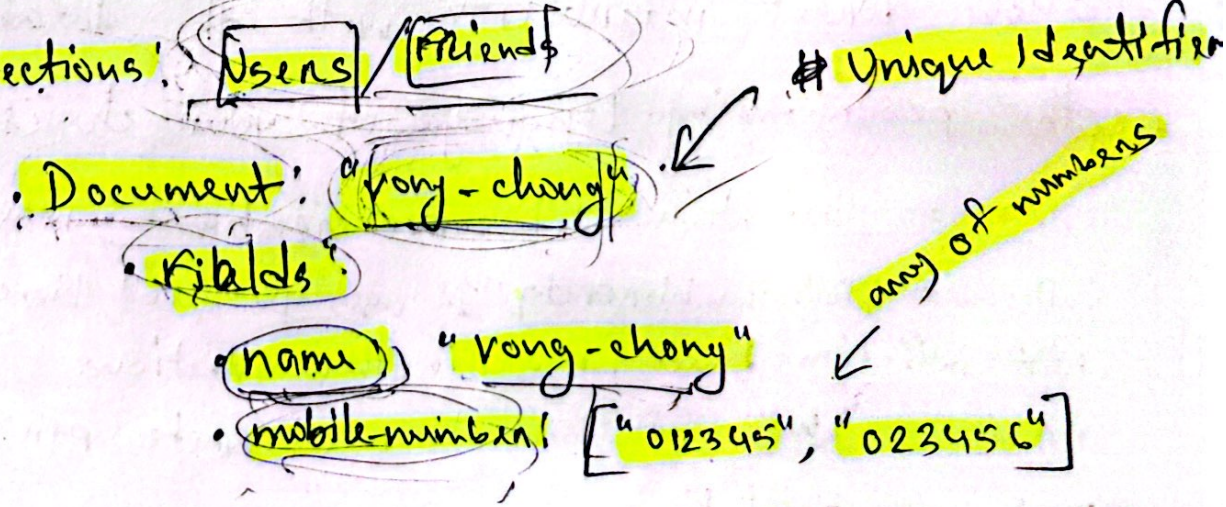
Firestore provides a **rich set of features**, including **real-time data synchronization**, **offline support**, **automatic scaling** and **powerful querying**

capabilities. It allows you to easily add, update and retrieve data of clients & libraries and SDKs.

Firestore does not use a specific database language. Instead, it provides clients libraries and SDKs for various programming languages, such as JavaScript, Java, Python and more.

Firestore API uses a combination of REST and RPC (Remote Procedure Call) protocol to communicate with the database backend

Example: "rong-chong" has two mobile numbers how you will store this in firestore?



powerful queries: complex query, ordering (asc, des), limit(docs) pagination.

How firestore scale the data?

⇒ Firestore provide automatic horizontal scalability. Ex? which means Firestore can handle increasing amount of data and traffics.

Firestore achieves scalability through following mechanism:

Data distribution, Automatic scaling, Indexing, Caching, offline support.

key characteristics (Data sharding, Distributed Load balancing, Automatic scaling)

DAICO

Firestore automatically manages the infrastructure and resources required to scale your database.

It allows you to focus on developing your application without worrying about scalability concerns.

Firestore Native method: The native methods refer to the methods provided by the firestore SDKs that allow direct interaction with the database using the programming language of your choice.

Firestore limitation: 1. Document size limit: max size limit 1MB

2. Document & Collection hierarchy: It uses Doc & Coll hierarchy. you can't nest collections directly within other collections.

3. Maximum number of documents Read/write per second (limit)

4. Indexing limitations

Where can we use firestore? 1. web & mobile app 2. CMS 3. IOT 4. Serverless app

FAAS/IAAS/CAAS/BAAS/PAAS

Key characteristic of Firestore:

→ Real-time data synchronization → if any update is made to the data it will imidiate reflected all the connected nodes.

→ offline supports: It provide offline support. like if when application is online if connected to the firestore DB. It will automatically create one local copy for user/client system. when application is offline, the client device can read and write the data from local copy. Firestore provides a local cache on the client, when it store the data if any changes made to the data in offline. Once the device reconnects to the internet, then Firestore automatically synchronizes the local changes.

→ powerful querying capability: It can perform the complex query by using multiple conditions. It allows you to use 'AND' and "OR" operation. It allows you to order query result by ascending and descending order. you can limit the number of documents returned by a query. using limit(). It provides pagination.

~~FAAS~~ ~~FAAS~~

~~It~~

Firestore native mode and data store mode:

Native mode: It is a scalable NOSQL Document database. introduced in 2019. It offers real-time data synchronization and automatic offline support. for web, mobile and server application. It provides a flexible data model, allowing you to store, query and organize your data in collections and documents. It supports powerful ~~post~~ querying capabilities, strong consistency. It integrates well with other firebase services, such as firebase Authentication, Cloud functions and cloud storage.

Datastore Mode: ~~Datab~~ It is also a scalable NOSQL document database. It also have a basic querying capabilities. It doesn't have ~~featu~~ rich-feautures like native mode ~~is~~ such as offline support, realtime data synchronization, advance querying.

FAAS/IAAS/CAAS/BAAS/PAAS

What is FAAS ? Example ? ✓
What is IAAS ? Example ? ✓
What is CAAS ? Example ? # What is SAAS ? Ex ? ✓
What is BAAS ? Example ?
What is PAAS ? Example ? ✓
Difference between faas & Iaas ?
Difference between FaaS & Paas ?
Why don't we use any functions for decopling ?
Hybrid Architecture with any faas & DB ? ✓
How database integration is done in cloud ?
Firestore native and datastore mode ?
Benefits for using faas framework ? ✓
In Faas, which one we use ! synchronous or asynchonous ? why ✓
Difference between Iaas & Paas ?
What is cloud computing ? ✓

FaaS: FAAS stands for Function as a Service. It is a cloud computing model where developers can write and deploy individual functions or units of code that perform specific tasks or operations, without the need to manage the underlying infrastructure. Example: AWS Lambda, Google cloud function, Azure function.

Benefits of using FaaS:

1. Simplified Development: It simplify the development process by allowing developers to focus on writing code for specific functions or tasks. Developers don't need to worry about underlying infrastructure.
2. Scalability: Faas automatically handle the scaling of functions based on demand.
3. Cost effeciency: you need to pay only for actual usage.
4. Event-driven architecture.

In faas, when we will use synchronous function and when we should use Asynchronous functions?

→ In faas, both synchronous and asynchronous function can be used based on specific requirements:

Synchronous functions: Synchronous function is commonly used in faas when immediate responses or real-ti processing is needed. When a request is made to a synchronous function, the function executes and blocks further execution until it completes and returns a response. This type of function is suitable for tasks that require immidiate results, such as processing user input, performing calculation or generating a quick response to an API call.

Asynchronous functions: Asynchronous functions are used when the response time is not critical, and the processing can be in the background. It used for tasks that involve time-consuming operations, such as data processing, batch jobs or sending notifications.

FaaS: FAAS stands for

It is

cloud computing: Cloud computing refers to the delivery of computing ~~servi~~ resources, such as servers, storage, databases, networking, software and applications over the internet on a pay-per-use basis. It allows users to access and utilize these resources remotely, without the need of local infrastructure or hardware.

Benefits:

1. Scalibility: on demand scaling automatically.
2. Flexibility: it provides flexibility in terms of resource allowcation
3. Cost-efficiency: users only have to pay what they use.
4. ~~Read~~ Reliability and Availability: Cloud service providers typically offer high levels of reliability and availability.

Architecture with FAAS and a DB?

1. First we need to use a FaaS provider like AWS Lambda, Google cloud functions or Azure functions.
2. Use an API Gateway.
3. Then need to set up event triggers for FaaS functions.
4. Then we need to choose one database for the applicati such as Relational DB ~~or~~ such as MySQL, PostgreSQL or NoSQL DB such as MongoDB, DynamoDB. The choice depends on specific requirments.
5. Then need to define the logic for function to handle incoming events.
6. Use DB to store and retrieve data required by functi
7. Then function can interact with the DB

FaaS ⇒ Normal users use করে, normally কোন coding করতে হয় না.

PaaS ⇒ Developers use করে.

IaaS ⇒ System administrations use করে.

What is Paas? Example?

→ Paas stands for Platform as a service. It is a cloud computing model that provides a platform and environment for developers to build, deploy and manage applications without the complexity of infrastructure management. In Paas, the cloud service provider takes care of the underlying infrastructure, including servers, storage and networking, while developers focus on writing and deploying their applications.

Benefits:

1. Development environment: Paas provides a development environment that includes tools, libraries and frameworks for app development. It provides a platform where developers can write, test and debug their code.
2. Deployment & Scalibility: Paas platform provide deployment mechanism that intigrate the process of deploying application to the cloud. It supports automatic scaling. allow app to handle increased traffic and workload.

cost efficiency: It follows pay-as-you-go model. Users are charged based on their usage. Reduce infrastructure costs.

Example: Google App Engine, AWS Elastic Beanstalk.

What is IAAS? Example?

⇒ It is a form of cloud computing that delivers the fundamental compute, such as network and storage resources to the customer over the internet on demand. It provides underlying OS, security, networking and servers for developing apps.

⇒ pay-as-you-go.

Benefits:

virtualized infrastructure: Iaas providers offer virtualized computing resources including virtual machine, storage and networking.

scalibility: It allows users to scale their resource according to their needs.

pay-as-you-go:

Infrastructure management: It can manage the underlying infrastructure.

Example: AWS services Elastic Computecloud (EC2), Google cloud platform.

IAAS	PAAS	SAAS
Application	Application	Application
Data	Data	Data
Runtime	Runtime	Runtime
middleware	middleware	middleware
O.S	OS	OS
Virtualization	virtualization	virtualization
Servers	Servers	Servers
storage	storage	storage
networking	Networking	networking

##

FAAS	SAAS
Faas focuses on executing individual functions or code snippets in response to specific events or triggers.	saas is a fully functional software that are centrally hosted and accessed by the users over the internet.
2. Faas platform are primarily used by developers to build event-driven architecture.	saas applications are developed by saas service providers.
3. In Faas, developers have more control over the code and can customize	In saas, Users have limited control over customizing the functionality.
4. The target audience for Faas is developers or technical teams.	The target audience for saas is end-users. Because it is a ready to use application.

impact the availability and performance.

What is Baas? Example?

→ Baas stands for Backend-as-a-service. It is a cloud computing model that provides developers with a pre-built backend infrastructure to support the development of web and mobile applications

Benefit!

→ Baas platforms typically offer managed databases, allowing developers to store and retrieve data without having set up and manage their own database servers.

→ It provide user management capabilities, including user registration, authentication and authorization.

What is CaaS?

→ CaaS stands for Container as a service. It is a cloud computing model that provides a platform for deploying, managing and scaling containerized application.

Example: Docker, kubernetes.

Benefits:

→ It simplify the application development

→ you can scale your application ~~by adding~~ horizontally by adding or removing containers as needed.

→ It offer a built-in mechanism for high availability and fault tolerance. It can automatically restart failed container

→ Cost efficiency (pay-as-you-go)

Disadvantage:

→ It may required learning curve, especially for individuals who are new of containerization.

→ The platform can be complex to setup and configure.

→ If there is connectivity issues, it can be

Software as a service: It is a cloud computing in which software applications are provided via inter In saas, cloud providers takes care of the software and hardware. We don't even need to install the software in our machines. So, ~~it~~ we don't need to pay for the hardwares and softwares maintenance

Example: Gmail. → We ~~don't~~ ~~we~~ simply can use the services by creating account

Dropbox → ~~is~~ Same as Gmail., Google drive.

Benefits:

→ Accessible from anywhere.

→ scale up or scale down.

→ pay-as-you-go.

→ reduced time (we can use app directly from browser)

→ cost effective.

→ software are automatically updated.

spring.datasource

spring.datasource.prod

spring.datasource.url = jdbc:mysql://localhost:3306/testdb

" . " .prod.url = " " /proddb

H2 DB is a popular open-source, in-memory and embedded Java database.

Benefit:

→ Lightweight

→ inmemory

→ fast - high performance.

sPring Boot

What is spring?
what is springBoot?
springboot initializer and what are the steps? → spring boot starters
How you will create website using springboot?
How get/post request work in spring boot? explain!
How you will deal with concurrency? critical situation? bootle neck? parallelisation? How? explain?

Answers:

Spring: Spring is a popular open-source framework for building enterprise java applications. It provides a comprehensive programming and configuration model for developing robust and scalable applications.

Key features: Dependency injection, Inversion of control (IOC), Aspect oriented programming (AOP), Data access and Integration, MVC framework.

springboot: It is a sub-project of spring framework. It reduce the boilerplate code and configuration.

Key-features: Auto-configuration, standalone application, production ready features, starter dependencies.

Spring Boot starters: some commonly used starter

~~• spring-boot-starter: core~~

Spring boot starters are a set of ~~starters~~ pre-configured dependencies that provide some common libraries and configurations for specific functionalities in your spring boot applications

- spring-boot-starters: core starter, including auto-configuration support, logging and YAML.
- spring-boot-starter-web: starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container.
- spring-boot-starter-thymeleaf: starter for building MVC web applications using thymeleaf views
- spring-boot-starter-test: starter for testing springboot applications with libraries including junit, Hamcrest, Mockito.
- lombok: Java annotation library which helps to reduce boilerplate code.
- spring-boot-devtools: provides fast application restarts,

al machines.

'Concurrency'. It refers to the ability of a system to execute multiple tasks or processes concurrently. It means that multiple tasks can start, run and complete in overlapping time periods.

In the context of software Development, concurrency is allows different parts of a program to make progress independently and concurrently (simultaneously), rather than executing sequentially.

Benefits:

Improved performance: overall efficiency of a system can be increased.

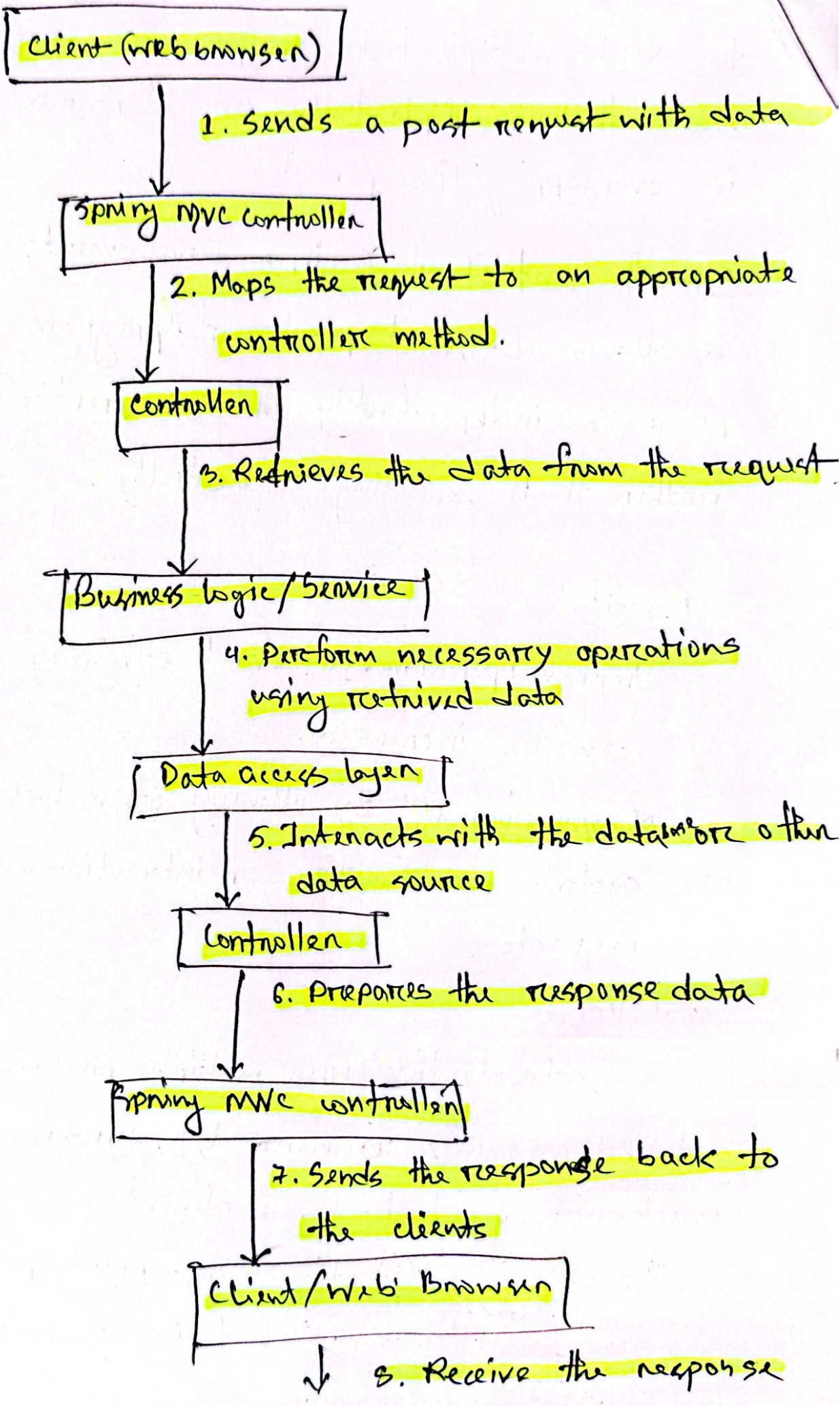
Responsiveness: By allowing other tasks to proceed, system can handle user interaction and process other requests.

challanges:

Synchronization: when multiple processes access shared resources concurrently, synchronization mechanism need to be implemented.

Debugging and testing: concurrent programs can be more challenging to debug and test compared to sequential programs.

Spring MVC → POST request?



... with concurrency?

There are some strategies and best practice for handling concurrency:

1. Synchronization: When multiple threads or processes shared resources, proper synchronization should be implemented to ensure data consistency.

2. Immutable objects: Design your objects to be immutable if possible. Because immutable objects cannot be modified once created. They can safely shared multiple threads without synchronization.

3. Asynchronous programming: Use asynchronous programming models to handle concurrency. This will allow tasks to run concurrently without blocking the execution flow

4. Testing and debugging: Thoroughly test your concurrent code to identify and fix concurrency-related issues.

5. Fine-Tuning: monitor and fine-tuning fine-tune your concurrent code to optimize performance and resource utilization.

Loose-coupling: To When any class is depend another class through its interface then we can achieve loose coupling.

For ex: WeatherProvider: provides the temp.

another class WeatherReporter it passes the instance for giving the temparature.

Dependancy injection: To design patten used in software development to achieve loose coupling and improve the modularity and testability:

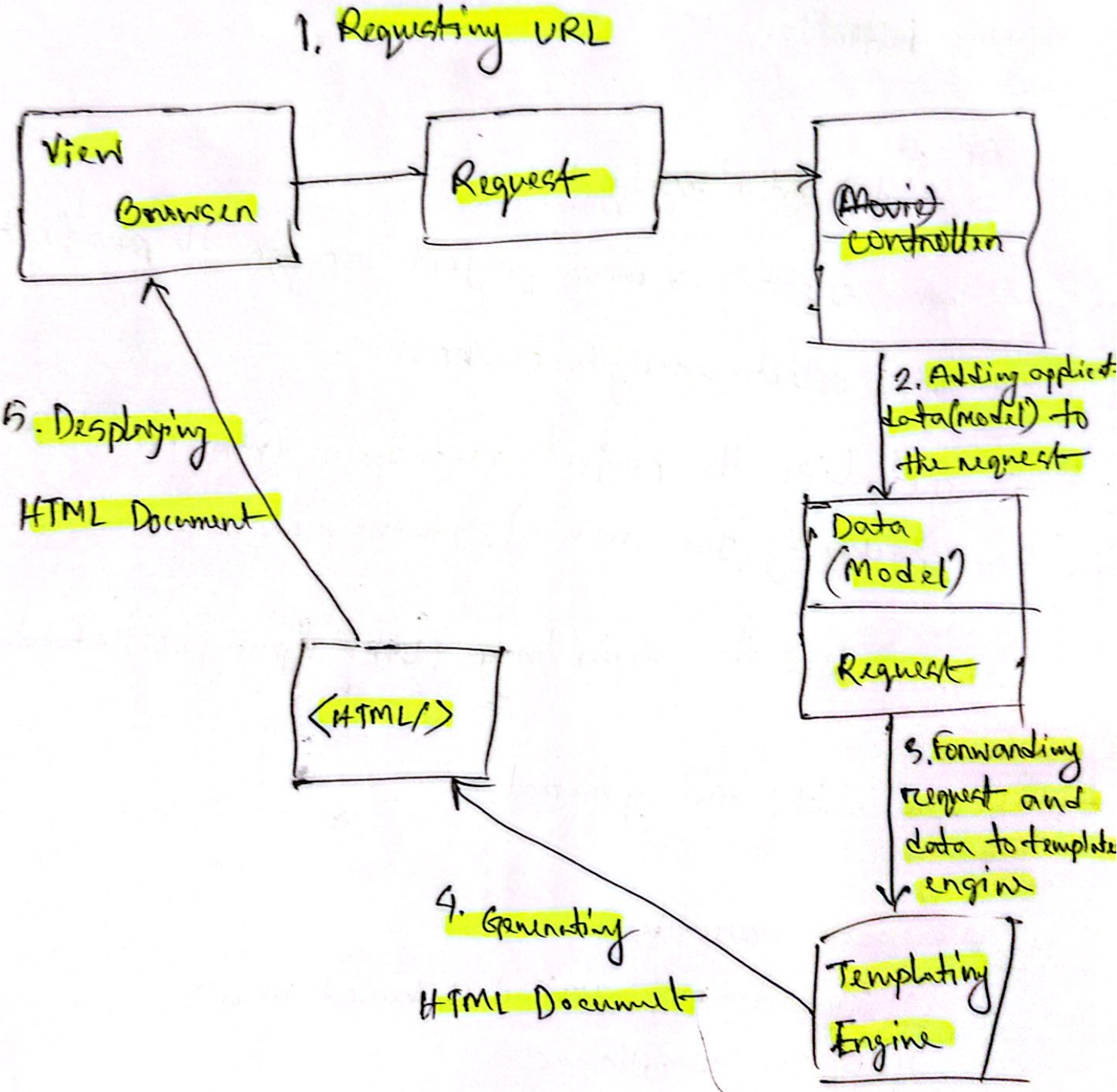
USDN

1

2

3

Spring MVC - How does this work in GET request?



LiveReload, and configurations for enhanced development experien

Spring Initializr:

→ go to start.spring.io

→ then select → maven project or gradle project

→ then select spring boot version.

→ initialize the project metadata. (name, Artifact, Descrip
packaging (Jar or War), Java version.

→ then add the dependency (like spring boot starters)

then click on generate.

H_2 database benefits:

1] In-memory and disk-based modes.

2] ACID compliance

3] Java integration

4] Lightweigh and embedded.

JPA

JPA
Details about JPA & key characteristics
How do you define a java class as a table in DB in JPA?
Why do you even use it?
What are the functionalities it offers?
Example about mappings in Table? [UNI + BIDIRECTIONAL MAPPING]
What are Repositories
Repository pattern
What is ORM?
What is Hibernate?
How do you implement db level relationships using JPA?
JPA Annotations?
Scenario about Rdbms (table structure, primary key, foreign key)?
Why we do normalization?
Why can't we store all our data in a single table rather keeping it multiple tables? [SB]
Benifit of H2 Database?
ONE TO ONE, ONEtoMany, ManyToMany with Example?
YAWNING AND INVERSION SIDE
How would you give some scenarios & ask which mapping? [SB]

ANSWERS

JPA: (JAVA PERSISTENCE API):
Its a set of java specifications for accessing, persisting, & managing data between java objects & relational Databases. In Spring Boot, JPA simplifies database accen & provides a smooth integration with the Framework.

Defining About JPA: JPA

1. JPA allows us to Map our java objects to Database tables & vice versa.
2. Instead of writing long SQL queries, JPA provides a way to manage our database operations (like CRUD) in an object manner (object-oriented)
3. When using Spring boot, the Framework takes care of the most of the boilerplate configuration for JPA, making it even easier to use.

KEY Characteristics:

ORM (object-relational Mapping): Maps JAVA objects to Database Table & vice versa
ANNOTATIONS: Uses Java ~~Annotations~~ Annotations (or XML) to define the mapping between classes & tables, fields & columns, etc.
Entity MANAGER: Central Component for API operations, It manages the lifecycle of entities (Data - where to keep, how to take etc.)
JPQL (JAVA PERSISTENCE QUERY LANGUAGE): A query language to perform database operations in a object-oriented way.

JPA (Java Persistence API)

PROVIDER AGNOSTIC: While JPA is a specification, there are Multiple implementations (HibernNate, EClipseLink etc), But Spring boot uses Hibernate as it default.

Integration with Spring data: Spring Boot can work well with SPRING Data JPA to further simplify database operations. allowing for easy repository creation, with the minimal boilerplate code.

Defining a Java Class As a Table in DB in JPA

It uses annotations: (For doing that) → ETIGECOMAONE

@Entity: Place this annotation at the top of the Java class. This tells JPA that this class is a database entity.

@Table(name = "your-table-name") specifies the name of the database table, if its not written, class name will be the table name.

@Id → Place this annotation on the field that represents the primary key of the table.

@GeneratedValue → often used with Id, indicates that the value of the primary key should be automatically generated

5. Others Annotations →

@Column(name = "Column-name") → specifies the column name in the DB for the annotated field. If not written, the field name is used.

Map Fields to column

@ManyToOne, @OneToMany → used for defining relationships between entities.

Establish Relationship

WHY DO WE USE JPA

We use JPA to make it easier to connect our Java programs with databases, instead of writing lengthy & complex data base code. JPA lets us work with our data in a more natural & Java friendly way, having a translator that lets JAVA & Databases speak the same language.

JPA (Java Persistence API)

What is JPA?
Details about JPA and key characteristic?
How dow you define a java class as a table in DB in JPA
Example about mappings between table?
What is uni and Bidirectional mapping?
ManyToOne, OneToOne, OneToMany, ManyToMany with example?
Why do we even use it?
What functionality does it offer?
What are repositories?
Repository pattern?
ORM (Object Relational Mapping)
Hibernate?
How do you implement DB level relationships using JPA?
Annotations?
RDBMS (Table structure, primary key, foreign key)
Why do we do normalisation?
Why can't we store all our data in a single table rather than keeping it in multiple tables?

* JPA: JPA stands for Java Persistence API. It is a programming interface that simplifies the way Java ~~program~~ applications interacts with Database. JPA provides a set of tools and techniques to handle the communication between the application and the database.

Key characteristics of JPA:

1. Object-Relational mapping (ORM): JPA provides a way to map Java object (entities) to relational DB tables, allowing developers to work with objects in their code while persisting and retrieving data from the DB.

2. Annotation-based mapping: JPA utilizes annotations (metadata) to define the mapping between Java classes and database tables. By adding annotations to entity classes, developers can specify how the fields and relationships of the objects are stored in DB.

3. CRUD operations: JPA simplifies the implementation of common DB operations, named CReate, Read, Update, Delete (CRUD). Developers can perform

ase operations on entities using JPA's EntityManger API. which handles the underlying SQL query and DB interactions.

Query Language: JPA provides the Java Persistence Query Language (JPQL), which is similar to SQL. JPQL allows developers to write queries using Java objects and their relationships.

5. Caching: JPA includes a caching mechanism that improves performance by storing frequently accessed data in memory. This reduces the number of DB queries needed to retrieve the data, resulting in faster application response time.

Repository pattern: It provides a structured approach to handle data access in an application. It increases code reusability, maintainability and testability.

* How do we define a java class as a table in Map

→ To define a java class as a table in a DB, table

JPA, we need to follow these steps :-

1. Annonate the class! Add the @Entity annotation to the java class that we want to map to a table. This annonation indicates that the class is an entity and should be persisted in the DB.

Example:

```
@Entity
public class Employee {
    // class fields, constructors, methods
}
```

2. Define Primary key: Specify the primary key for the entity using the @Id annotation on the corresponding field or property. This annotation marks the field as the primary key of the table.

Example:

```
@Entity
public class Employee {
    @Id
    private Long id;

    // - - - - - -
}
```

3. Map fields to column: Map the entity's fields to the table columns using annotations such as @Column, @JoinCol or @Embedded. These annotations ~~such as~~ specify the name, datatype and other properties of the columns.

Example:

```
@Entity
public class Employee {
    @Id
    private Long id;

    @Column(name = "employee_name")
    private String name;

    @Column(name = "employee_age")
    private int age;

    // other fields - - - -
}
```

4. Establish Relationships (if any): If your entities has relationships with other entities, such as one-to-on one-to-many or many-to-many, ~~so~~ then you need to define those relationships using annotations like @OneToOne, @OneToMany or @ManyToMany. These annotations specify the relationship type, target entity and other detail

Example:

```
@Entity
public class Employee {
    @Id
    private Long id;

    @Column(name = "employee_name")
    private String name;

    @OneToMany(mappedBy = "employee")
    private List<Task> tasks;

    // other fields - - - .

}
```

- Example about mappings between table?

  Lets say we have two entities: Book and Author. Each book have one author and each author can have multiple books. Now, lets define the mappings between these entities!

```
@Entity
@Table(name="books")
public class Book {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  private String title;

  @ManyToOne
  @JoinColumn(name = "author_id")
  private Author author;

  // constructor, getters, setters
}
```

1. @Entity annotation marks it as an entity class.

2. @Table annotation specifies the table name as "books"

3. @Id annotations indicates primary key of the table.

4. The @GeneratedValue annotation with GenerationType.IDENTITY strategy indicates that the primary key values are automatically generated.

5. The title field represents a column with the same name in the "books" table

6. The @ManyToOne annotation denotes a many-to-one relationship with the Author entity.

7. The @JoinColumn annotation specifies the foreign key column name as "author_id"

Example:

2. Author Entity:

```
@Entity
@Table(name = "authors")
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    @OneToMany(mappedBy = "author", cascade = CascadeType.ALL)
    private List<Book> books;

    // constructor, g, s
}
```

1. @Entity ann makes it as an entity class.
2. @Table ann specifies the table name as "authors".
3. @Id ann indicates the primary key of the table.
4. @GeneratedValue with @GT.ID indicates that the primary key values are automatically generated.
5. name field represents a column with the same name in "authors" table.
6. @OneToMany indicates a one-to-many relationship with Book entity.
7. mappedBy = "author" specifies the field in Book entity that owns the relationship.
8. cascade = CascadeType.ALL ensures that when an author is deleted, all associated books are also deleted.

Table: books

id	title	author_id
1	Title 1	1
2	Title 2	1
3	Title 3	2

Table: authors

id	name
1	Author 1
2	Author 2

What is Unidirectional and Bidirectional Mapping?

Unidirectional Mapping: In a unidirectional mapping, the relationship between two entities is defined in one direction. This means that one entity has a reference to another entity, but the referenced entity does not have a direct association to the first entity.

For example: The Book entity has a reference of Author entity, but ~~the referenced entity doA~~ Author entity does not have a direct reference to Book entity.

Bidirectional Mapping: In a bidirectional mapping, the relationship between two entities is defined in both directions. That means each entity has a reference to the other entity.

For example: a bidirectional mapping between Book and Author entities means that Book entity has a reference to the Author entity, and Author entity has a reference to the corresponding Book entity.

```
@Entity
public class Book {
    // other fields and annotations

    @ManyToOne
    @JoinColumn(name = "author_id")
    private Author author

    // getters, setters
}
@Entity
public class Author {
    ---
    @OneToMany(mappedBy = "author")
    private List<Book> books
{   ---
```

To One: Example: One person can have one id/passport and each passport/id belongs to one person.

OneToMany: Example: An Author can have multiple books.

OneToMany relationship with Book.

ManyToMany: Example: An student entity can have many-to-many relationship with course entity. That indicating that a student can enroll in multiple courses. and course can have multiple students.

One-to-One relationship Functionality:

→ Data Integrity, Object Navigation, Shared Primary key

One-to-many → Functionality

→ parent child relationship, Lazy loading and Eagen fetching.

Many-to-many → Functionality

→ Bidirectional association, join table, Quining and filtering.

Hibernate ?

⇒ Hibernate is an open source object-relational Mapping (ORM) framework for java. It provides an convinient and efficient way to map java object to the relational database table. It allowing developers to work with database using object-oriented programming concepts.

Why do we can't we store all our data in a single table rather than multiple tables ?

→ ~~For~~ Reduce data redundancy

→ Data consistency

→ improve query performance

→ Flexibility and scalibility.