SOAP – Simple Object Access Protocol

- Simple Object Access Protocol(SOAP) is a network protocol for exchanging structured data between nodes. Because it is a protocol, it imposes built-in rules that increase its complexity and overhead, which can lead to longer page load times. However, these standards also offer built-in compliances that can make it preferable for enterprise scenarios. The built-in compliance standards include security, atomicity, consistency, isolation, and durability (ACID), which is a set of properties for ensuring reliable database transactions.
- It uses XML format to transfer messages. It works on top of application layer protocols like HTML and SMTP for notations and transmission. SOAP allows processes to communicate throughout platforms, languages and operating systems, since protocols like HTTP are already installed on all platforms.
- SOAP is a light weight data interchange protocol because it is based on XML. designed to be OS and Platform independent. built on top of HTTP. SOAP is mainly used for Web Services and Application Programming Interfaces (APIs).

Fenil Shah

SOAP Message Format

- SOAP message transmits some basic information as given below
  - Information about message structure and instructions on processing it.
  - Encoding instructions for application defined data types.
  - Information about Remote Procedure Calls and their responses.
  - The message in XML format contains three parts
- **Envelope**: It specifies that the XML message is a SOAP message. A SOAP message can be defined as an XML document containing header and body encapsulated in the envelope. The fault is within the body of the message.
- **Header**: This part is not mandatory. But when it is present it can provide crucial information about the applications.
- **Body**: It contains the actual message that is being transmitted. Fault is contained within the body tags.
- **Fault**: This section contains the status of the application and also contains errors in the application. This section is also optional. It should not appear more than once in a SOAP message.

SOAP Message Format

```
Content-Type: application/soap+xml

<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
        <env:Header>
                <m:GetLastTradePrice xmlns:m="Some-URI" />
        </env:Header>
        <env:Body>
                <symbol xmlns:p="Some-URI" >DIS</symbol>
        </env:Body>
</env:Envelope>
```

Fenil Shah

SOAP vs REST

- REST is a set of guidelines that offers flexible implementation, whereas SOAP is a protocol with specific requirements like XML messaging.
- REST APIs are lightweight, making them ideal for newer contexts like the Internet of Things (IoT), mobile application development, and serverless computing. SOAP web services offer built-in security and transaction compliance that align with many enterprise needs, but that also makes them heavier. Additionally, many public APIs, like the Google Maps API, follow the REST guidelines.
- SOAP uses only XML for exchanging information in its message format whereas REST is not restricted to XML and its the choice of implementer which Media-Type to use like XML, JSON, Plain-text. Moreover, REST can use SOAP protocol but SOAP cannot use REST.
- SOAP is difficult to implement and it requires more bandwidth whereas REST is easy to implement and requires less bandwidth such as smartphones.
- Benefits of SOAP over REST as SOAP has ACID compliance transaction. Some of the applications require transaction ability which is accepted by SOAP whereas REST lacks in it.
- On the basis of Security, SOAP has SSL(Secure Socket Layer) and WS-security whereas REST has SSL and HTTPS. In the case of Bank Account Password, Card Number, etc. SOAP is preferred over REST. The security issue is all about your application requirement, you have to build security on your own. It’s about what type of protocol you use.

Fenil Shah

REST API

- if you want to interact with a computer or system to retrieve information or perform a function, an API helps you communicate what you want to that system so it can understand and fulfill the request. a mediator between the users or clients and the resources or web services they want to get. It's also a way for an organization to share resources and information while maintaining security, control, and authentication—determining who gets access to what.
- Representational State Transfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. REST API is a way of accessing web services in a simple and flexible way without having any processing.
- REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST uses less bandwidth, simple and flexible making it more suitable for internet usage. It's used to fetch or give some information from a web service. All communication done via REST API uses only HTTP request.
- When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text. JSON is the most generally popular file format to use because, despite its name, it's language-agnostic, as well as readable by both humans and machines.

Fenil Shah

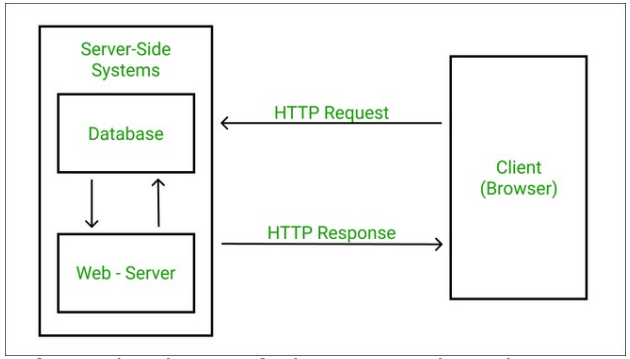
RESTful API

- A client-server architecture made up of clients, servers, and resources, with requests managed through HTTP.
- Stateless client-server communication, meaning no client information is stored between get requests and each request is separate and unconnected.
- Cacheable data that streamlines client-server interactions.
- A uniform interface between components so that information is transferred in a standard form. This requires that:
  - resources requested are identifiable and separate from the representations sent to the client.
  - resources can be manipulated by the client via the representation they receive because the representation contains enough information to do so.
  - self-descriptive messages returned to the client have enough information to describe how the client should process it.
  - hypertext/hypermedia is available, meaning that after accessing a resource the client should be able to use hyperlinks to find all other currently available actions they can take.
- A layered system that organizes each type of server (those responsible for security, load-balancing, etc.) involved the retrieval of requested information into hierarchies, invisible to the client.

Fenil Shah

HTTP Request

- HTTP (Hypertext Transfer Protocol) specifies a collection of request methods to specify what action is to be performed on a particular resource. An HTTP request is made by a client, to a named host, which is located on a server. The aim of the request is to access a resource on the server. To make the request, the client uses components of a URL (Uniform Resource Locator), which includes the information needed to access the resource.
- A correctly composed HTTP request contains the following elements:
  - A request line.
  - A series of HTTP headers, or header fields.
  - A message body, if needed.



- Each HTTP header is followed by a carriage return line feed (CRLF). After the last of the HTTP headers, an additional CRLF is used (to give an empty line), and then any message body begins.
- **Request line**: The request line is the first line in the request message. The method is a one-word command that tells the server what it should do with the resource. The path component of the URL for the request. The path identifies the resource on the server. The HTTP version number, showing the HTTP specification to which the client has tried to make the message comply.
- E.g. - GET /software/htp/cics/index.html HTTP/1.1

Fenil Shah

HTTP Request

- **HTTP headers**: HTTP headers are written on a message to provide the recipient with information about the message, the sender, and the way in which the sender wants to communicate with the recipient. Each HTTP header is made up of a name and a value. The HTTP protocol specifications define the standard set of HTTP headers, and describe how to use them correctly.
- RequestHeaders are sent from frontend to backend. Request headers contain more information about the resource to be fetched, or about the client requesting the resource
- In the request header, you enter the key value of the application. The two main key values are given below.
- Content-Type: The format of data is specified by Content-Type. Mainly developers use JSON format in the content type.
- Authorization: This information is included to identify the requester
- **Message body**: message body contains the entity body, which can be in its original state, or can be encoded in some way for transport, such as by being broken into chunk. Message bodies are appropriate for some request methods and inappropriate for others. For example, a request with the POST method, which sends input data to the server, has a message body containing the data. A request with the GET method, which asks the server to send a resource, does not have a message body.

Fenil Shah

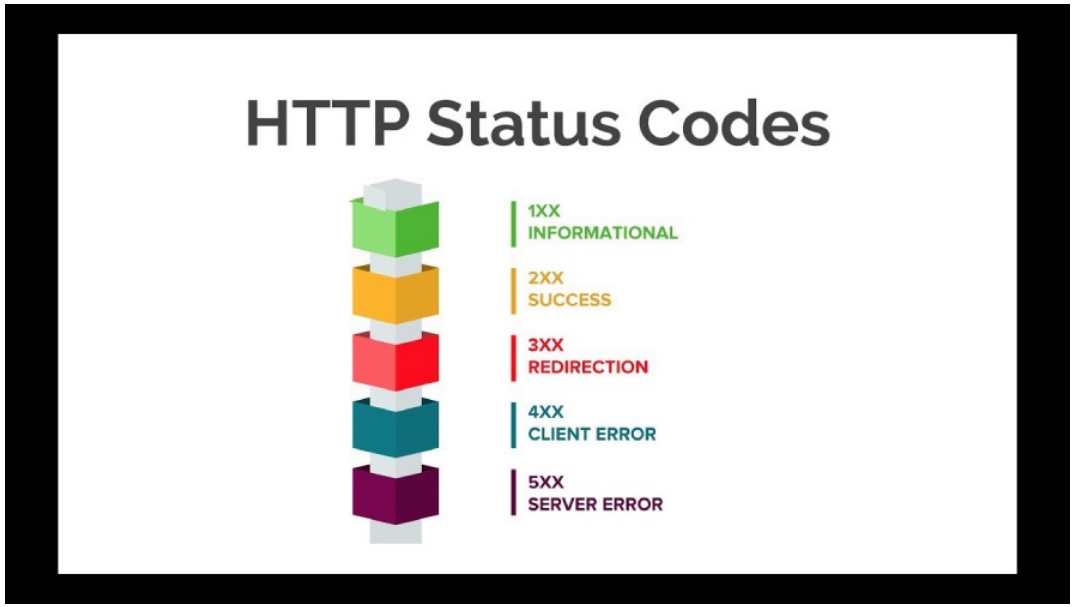
HTTP Request

- **HTTP protocol**: The correct format for HTTP requests and responses depends on the version of the HTTP protocol (or HTTP specification) that is used by the client and by the server. The versions of the HTTP protocol (or "HTTP versions") commonly used on the Internet are HTTP/1.0.
- **HTTP responses**: An HTTP response is made by a server to a client. The aim of the response is to provide the client with the resource it requested, or inform the client that the action it requested has been carried out; or else to inform the client that an error occurred in processing its request. Response headers are sent from backend to frontend. Response headers hold additional information about the response, like its location or about the server providing it.
- **Request Methods**:
- **GET**: This method retrieves information from the given server using a given URI. GET request can retrieve the data. It cannot apply other effects on the data.
- **POST**: The POST request sends the data to the server. For example, file upload, customer information, etc. using the HTML forms.
- **PUT**: The PUT method is used to replace all the current representations of the target resource with the uploaded content.
- **DELETE**: The DELETE method is used to remove all the current representations of the target resource, which is given by URI.

Fenil Shah

HTTP Status Codes

- The 1xx (Informational) class of status code indicates an interim response for communicating connection status or request progress prior to completing the requested action and sending a final response.
- The 2xx (Successful) class of status code indicates that the client's request was successfully received, understood, and accepted.
- The 3xx (Redirection) class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request.
- The 4xx (Client Error) class of status code indicates that the client seems to have erred.
- The 5xx (Server Error) class of status code indicates that the server is aware that it has erred or is incapable of performing the requested method.



HTTP Status Codes

- 200– For successful request.
- 201– For successful request and data was created
- 204– For Empty Response
- 400– For Bad Request. Server is expecting something and Frontend is sending something else .
- 401– For Unauthorized access. Authentication failed or the user does not have permission for the requested operation.
- 403– For Forbidden, Access Denied (User is authenticated but unauthorized)
- 404– For data not found.
- 405– For method not allowed or requested method is not supported. Like you have created an api for post but you are request in the browser as
- 500– Internal server error.
- 503– For Service unavailable

Postman

- Postman sends the request to the webserver and then the server sends the response back to it. A user has to set all the headers and cookies API expects to check the response.
- You can install the postman from the link Postman. This tool provides a collection of API calls, and you need to follow these API calls for testing APIs of the application. You will find a dropdown list with multiple methods.
- **cURL**: It stands for “client URL” and is used in command line or scripts to transfer data. It is a great tool for dealing with HTTP requests. Check curl –version. E.g - curl --location --request GET http://localhost:9090/v1/pathVariable/2

- **@Autowired**: By declaring all the bean dependencies in a Spring configuration file, Spring container can autowire relationships between collaborating beans. This is called Spring bean autowiring, done using @SpringBootApplication. It tells spring to pick the reference of class from the IOC container. Spring uses the bean's name as a default qualifier value. It will inspect the container and look for a bean with the exact name as the property to autowire it. Spring cannot resolve a bean for wiring, it will throw an exception. It can't be used for primitive and string values.

- **@Primary**: indicates that a bean should be given preference when multiple candidates are qualified to autowire a single-valued dependency. when you find more than one beans that both can be autowired, please use the primary one as your first choose.

- **@Qualifier**: indicates specific bean should be autowired when there are multiple candidates. No matter how many beans you've found, just use the one I tell you. @Qualifier is more specific and has high priority. So when both @Qualifier and @Primary are found, @Primary will be ignored.

- **@RequestMapping**: used to map web requests to Spring Controller methods. @GetMapping,@PostMapping, etc are added to improve the readability and reduce the verbosity of the code.

```
@RequestMapping(
  value = "/ex/foos",
  method = GET,
  produces = { "application/json", "application/xml" }
)
```

- **@RequestParam**: to extract query parameters, form parameters, and even files from the request. @RequestParam(required = false) . @RequestParam(defaultValue = "test"). @RequestParam annotation can specify default values if a query parameter is not present or empty by using a defaultValue attribute, provided the required attribute is false

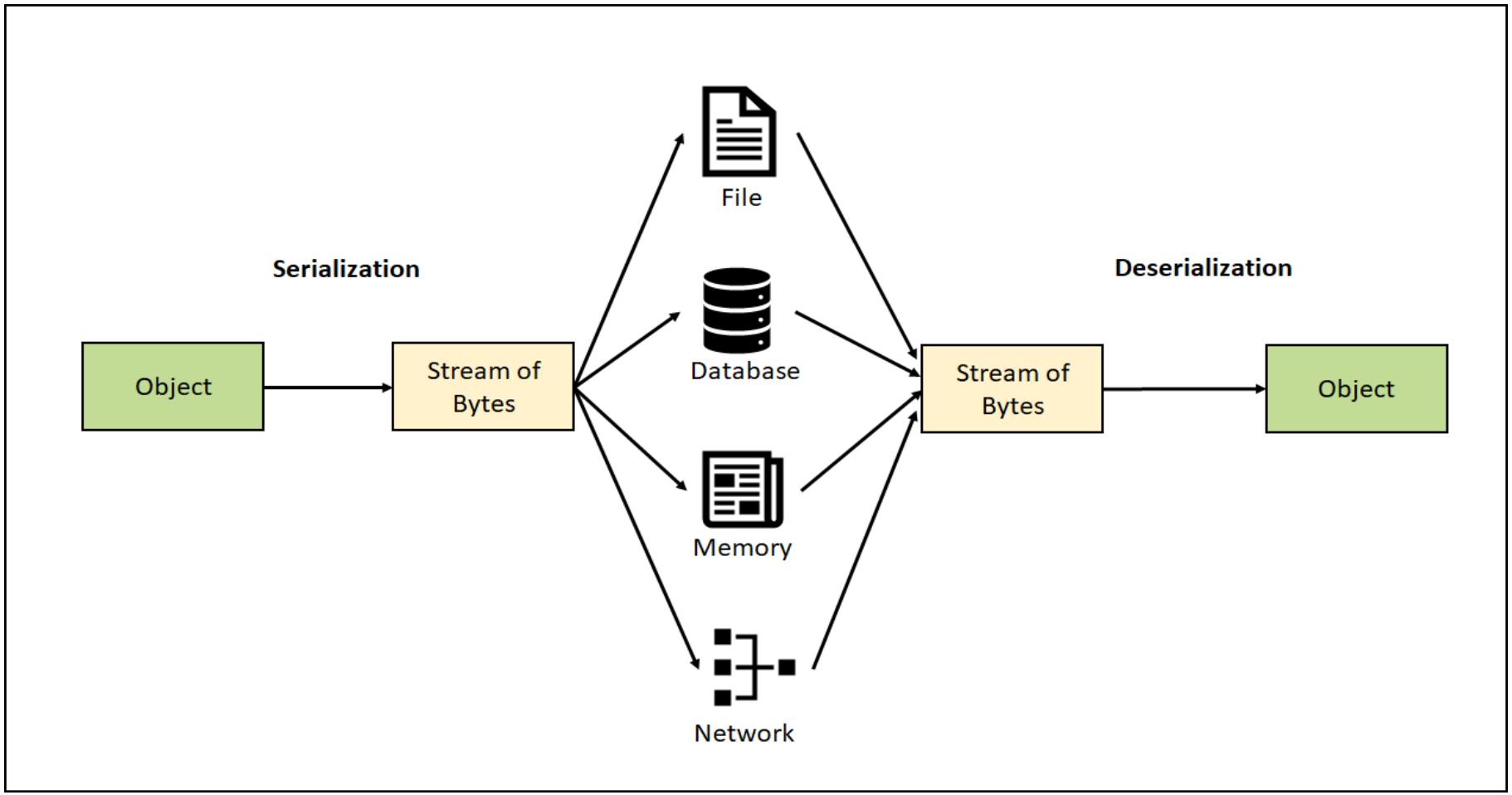
- **@PathVariable**: can be used to handle template variables in the request URI mapping, and set them as method parameters. Specify Path Variable with name eg- @PathVariable("id") String employeeId. @PathVariable Optional<String> id.
- @RequestParam is more useful on a traditional web application where data is mostly passed in the query parameters while @PathVariable is more suitable for RESTful web services where URL contains values. @PathVariable is extracting values from the URI path, it's not encoded. On the other hand, @RequestParam is encoded.

- **@RequestBody**: Maps the HttpRequest body to a transfer or domain object, enabling automatic deserialization of the inbound HttpRequest body onto a Java object. @ResponseBody annotation tells a controller that the object returned is automatically serialized into JSON and passed back into the HttpResponse object. we don't need to annotate the @RestController-annotated controllers with the @ResponseBody annotation since Spring does it by default.

- **@Configuartion**: 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.

- **@Bean**: Applied on a method to specify that it returns a bean to be managed by Spring context. Spring Bean annotation is usually declared in Configuration classes methods. Eg - What if we want Bean of class coming from Spring?

Serialization - Deserialization



Serialization - Deserialization

- Imagine you are talking to your friend over a cellular network. Your voice is audible to your friend who is sitting miles away from your home. You may also know that your voice gets converted into electrical signals which are then propagated to a mobile tower via radio signals. The tower then channels the electrical signal to the specific phone number you are in a connection with.
- Your voice goes through a lot of conversions to reach your friend on the other side.
- Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.
- The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform. To save/persist state of an object. To travel an object across a network. Deserialization requires less time to create an object than an actual object created from a class. hence serialization saves time. Serialization helps to implement persistence in the program. It helps in storing the object directly in a database in the form of byte streams. This is useful as the data is easily retrievable whenever needed.

Fenil Shah

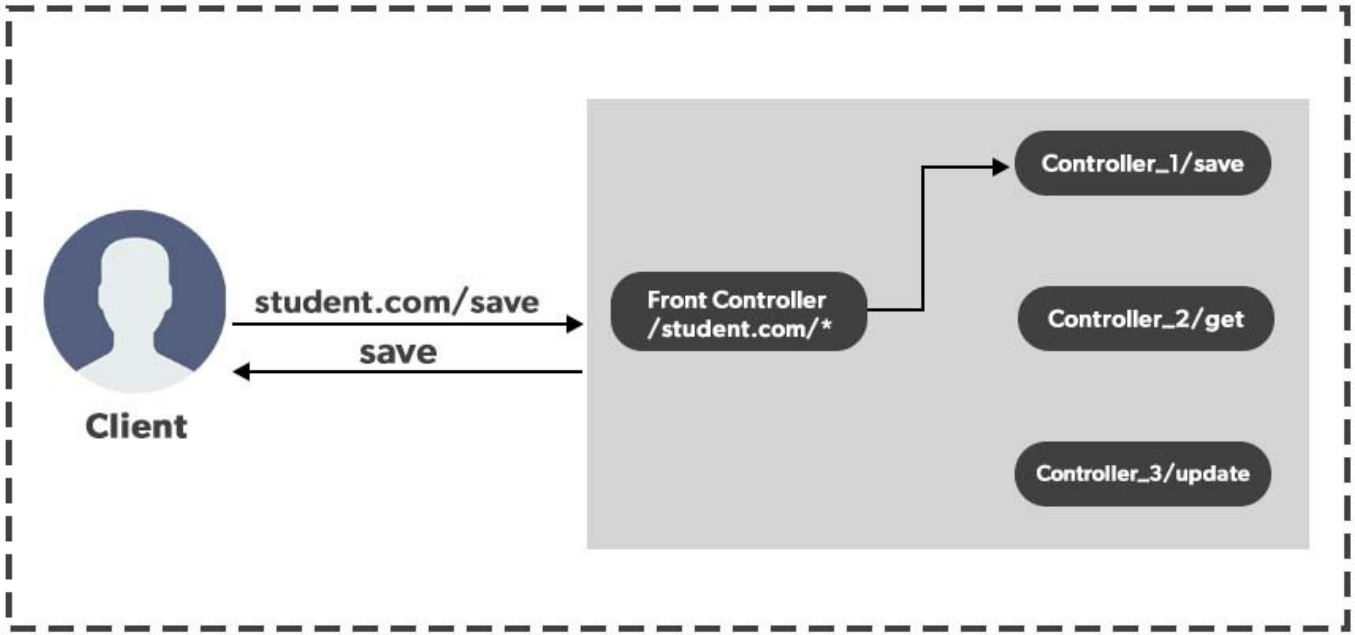
Serialization - Deserialization

- This version UID is unique for every object. Every class which implements the Serializable interface gets a unique Serial Version ID. This is useful to check whether the sender and receiver of a serialized object is loading the same class or not. If the UID's do not match, then the compiler throws an InvalidClassException.
- Only the objects of those classes can be serialized which are implementing java.io.Serializable interface. Serializable is a marker interface (has no data member and method). It is used to "mark" java classes so that objects of these classes may get certain capability. Constructor of object is never called when an object is deserialized.
- @RequestBody : Converts JSON to javsa Object (Deserialization)
- @ResponseBody : Converts Java Object to JSON and returns to client (Serialization)
- A variable defined with transient keyword is not serialized during serialization process.This variable will be initialized with default value during deserialization. (e.g: for objects it is null, for int it is 0)
- A variable defined with static keyword is not serialized during serialization process.This variable will be loaded with current value defined in the class during deserialization.

Fenil Shah

Dispatcher Servlet

- DispatcherServlet acts as the Front Controller for Spring-based web applications. For example, refer to the below image. Suppose we have a website called student.com and the client is make a request to save student data by hitting the following URL student.com/save and its first come to the front controller and once the front controller accepts that request it is going to assign to the Controller_1 as this controller handle the request for /save operation. Then it is going to return back the response to the Client.
- DispatcherServlet handles an incoming HttpRequest, delegates the request, and processes that request according to the configured HandlerAdapter interfaces that have been implemented within the Spring application along with accompanying annotations specifying handlers, controller endpoints, and response objects.
- the front controller is already created by the Spring Framework Developer, and the name of that particular controller is DispatcherServlet. You can use that front controller in your Spring MVC project. You really not required to create a front controller but you can reuse that front controller created by the Spring Framework Developer and they named it as DispatcherServlet.


Client
student.com/save
save
Front Controller
/student.com/*
Controller_1/save
Controller_2/get
Controller_3/update

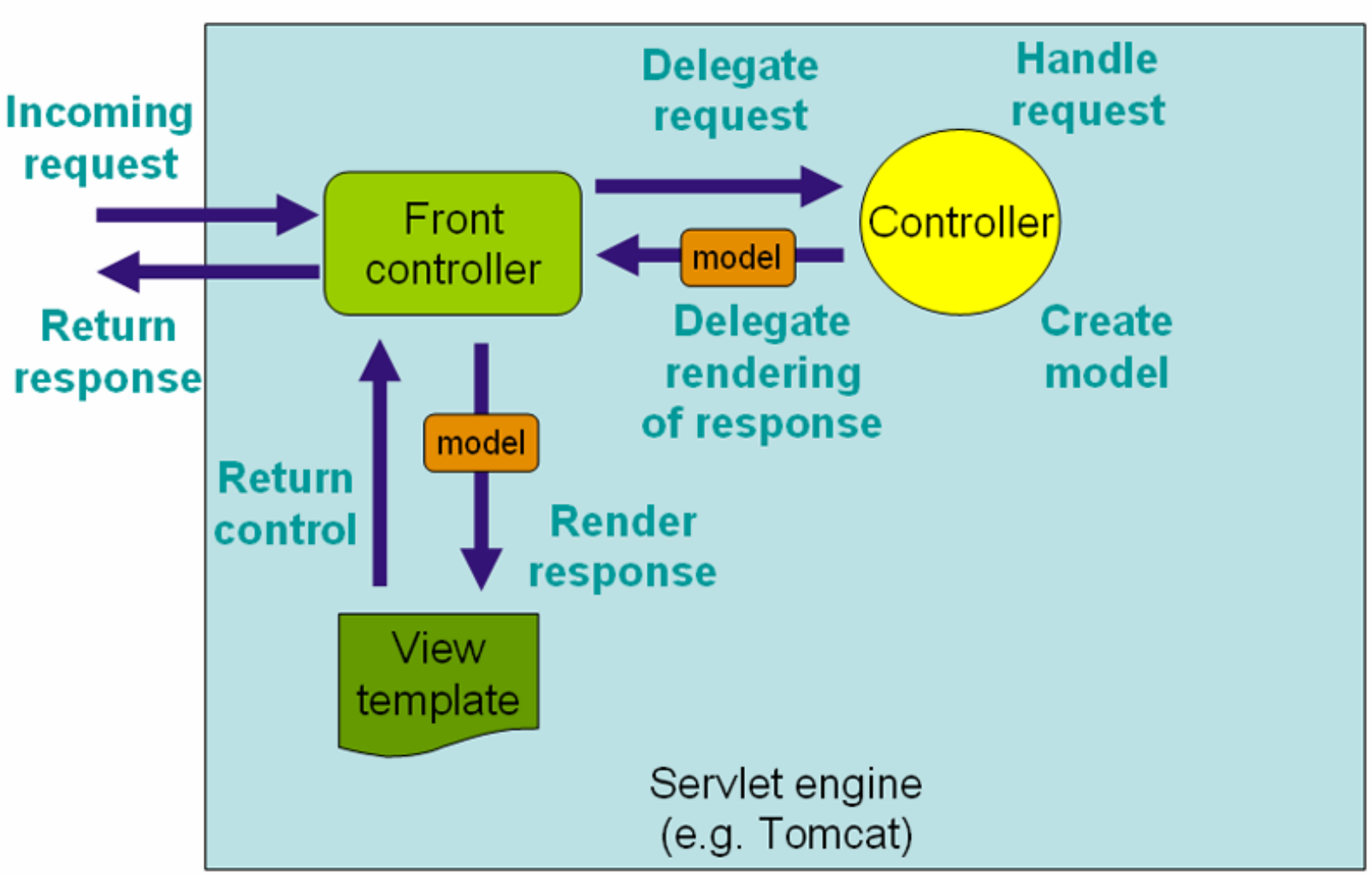
Fenil Shah

LOMBOK

- Project Lombok is a java library that automatically plugs into your editor and build tools, spicing up your java. Never write another getter or equals method again, with one annotation your class has a fully featured builder, Automate your logging variables, and much more. Eg - https://projectlombok.org/features/
- We can see the Getter and Setter functions added using Lombok In the compiled classes inside target folder. Getters, Setters and all lombok Code is created fine only . We need to install the plugin in IDE so that IDE recognises the code created by Lombok
- If I publish my code written using Lombok , will the person who clones it be able to run it without the Lombok plugin installed in IDE – No
- Both plugins and dependencies are Jar files. But the difference between them is, most of the work in maven is done using plugins; whereas dependency is just a Jar file which will be added to the classpath while executing the tasks.
- For example, you use a compiler-plugin to compile the java files. You can't use compiler-plugin as a dependency since that will only add the plugin to the classpath, and will not trigger any compilation. The Jar files to be added to the classpath while compiling the file, will be specified as a dependency.

SPRING MVC

- Spring MVC Framework follows the Model-View-Controller design pattern. It is used to develop web applications. It works around DispatcherServlet.
- DispatcherServlet handles all the HTTP requests and responses. It dispatches the requests to handlers. It uses @Controller and @RequestMapping as default request handlers. The @Controller annotation defines that a particular class is a controller.
- Spring Boot provides the spring-boot-starter-web library for developing web applications using Spring MVC. The Spring Boot autoconfiguration registers and configures the DispatcherServlet automatically. Therefore, we don’t need to register the DispatcherServlet manually.
- @RequestMapping annotation maps web requests to Spring Controller methods. The terms model, view, and controller are:
- **Model**: The Model encapsulates the application data.
- **View**: View renders the model data and also generates HTML output that the client’s browser can interpret.
- **Controller**: The Controller processes the user requests and passes them to the view for rendering.


Incoming request
Return response
Front controller
Delegate request
Handle request
Controller
model
Delegate rendering of response
Create model
Return control
model
Render response
View template
Servlet engine (e.g. Tomcat)

SPRING MVC

- Spring MVC Framework works as follows:
- All the incoming requests are intercepted by the DispatcherServlet that works as the front controller. The DispatcherServlet then gets an entry of handler mapping from the XML file and forwards the request to the controller.
- The object of ModelAndView is returned by the controller.
- The DispatcherServlet checks the entry of the view resolver in the XML file and invokes the appropriate view component..
- **Advantages**:
- The container is used for the development and deployment of applications and uses a lightweight servlet. It enables rapid and parallel development. Development of the application becomes fast. Easy for multiple developers to work together. Easier to Update the application. It is Easier to Debug because we have multiple levels in the application.
- **Disadvantages** : It has high complexity to develop the applications using this pattern. It is not suitable for small applications which affect the application's performance and design.

In Memory Storage VS Disk Storage

- The term "memory" usually means RAM (Random Access Memory); RAM is hardware that allows the computer to efficiently perform more than one task at a time (i.e., multi-task). The terms "disk space" and "storage" usually refer to hard drive storage. Hard drive storage is typically used for long-term storage of various types of files. Higher capacity hard drives can store larger amounts and sizes of files, such as videos, music, pictures, and documents. This shows that RAM memory of the server is very less and thus not feasible for data storage of your system .
- All servers will be communicating with a single source of truth where data won't be lost on stopping the server .
- Speed as when we used HashMap for storage , we were fetching from the main memory of your server which is very fast when compared to network calls from IP1 to IP2 . But still we use databases for storage and not server main memory because in the latter case , Correctness will be compromised for optimisation which is not desirable .
- To overcome slowness in case of disk (database) , we can use cache .

Fenil Shah

Need for DBMS

- A Data Base Management System is a system software for easy, efficient and reliable data processing and management. It can be used for:
  - Creation of a database.
  - Retrieval of information from the database.
  - Updating the database.
  - Managing a database.
- **Processing Queries and Object Management**: We can directly store data in the form of objects in a database management system. Application level code needs to be written to handle, store and scan through the data in a file system whereas a DBMS gives us the ability to query the database.
- **Controlling redundancy and inconsistency**: A database system provides redundancy control whereas in a file system, same data may be stored multiple times. For example, if a student is studying two different educational programs in the same college, say ,Engineering and History, then his information such as the phone number and address may be stored multiple times, once in Engineering dept and the other in History dept. Therefore, it increases time taken to access and store data. This may also lead to inconsistent data states in both places. A DBMS uses data normalization to avoid redundancy and duplicates.

Fenil Shah

Need for DBMS

- **Efficient memory management and indexing**
- **Concurrency control and transaction management**: DBMS provides mechanisms to deal with this kind of data inconsistency while allowing users to access data concurrently. A DBMS implements ACID(atomicity, durability, isolation,consistency) properties to ensure efficient transaction management without data corruption.
- **Access Control and ease in accessing data**: DBMS can grant access to various users and determine which part and how much of the data can they access from the database thus removing redundancy. Users can specify exactly what they want to extract out of the data.
- **Multiple User Interface**
- **Data scalability, expandability and flexibility**: We can change schema of the database, all schema will be updated according to it.
- **Security**: Simplifies data storage as it is possible to assign security permissions allowing restricted access to data.

Fenil Shah

RELATIONAL DATABASE

- Structured. stores data in the form of a table. The table consists of rows and columns, and in a relational database rows are referred to as records, and columns are referred to as fields.
- Store data in rows and columns like a spreadsheet.
- Each table has a special column that contains only distinct and unique values that are called primary key. This primary key is used to define the relationship between the tables.
- Popular way of interacting with a relational database is SQL(Structural Query Language), which allows access, filter, and modify data. E.g. - MySQL, Oracle

NON-RELATIONAL DATABASE

- Semi-structured or unstructured. No-SQL databases, that doesn't require any table, fields, or records. data is stored in form of graphs, documents(XML, JSON).
- Rather than containing tables, it consists of files within various folders. They can possess any kind of data, whether JSON, XML, etc. So, creating and managing data in NoSQL is easy and faster.
- These types of databases can store both structured and unstructured, whereas relational databases can store only data in a structured way.
- Large Data & Cheap: These databases can scale to accommodate any type of data while maintaining a low cost. E.g. – MongoDB, Apache Cassandra.

Fenil Shah

RELATIONAL DATABASE

- There are limitless indexing capabilities, which results in faster query response times.
- Their models can ensure and enforce business rules at the data layer adding a level of data integrity not found in a non-relational database.

NON-RELATIONAL DATABASE

- They provide scalability and flexibility to meet changing business requirements.
- They are document oriented.

Relational Database

- **ACID Compliance**:
  - **Atomicity**: Each transaction is considered as one unit, and either runs to completion or is not executed at all
  - **Consistency**: If a transaction occurs and results in data that does not follow the rules of the database, it will be 'rolled back' to a previous iteration of itself (or 'state') which complies with the rules. This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database.
  - **Isolation**: Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.
  - **Durability**: This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

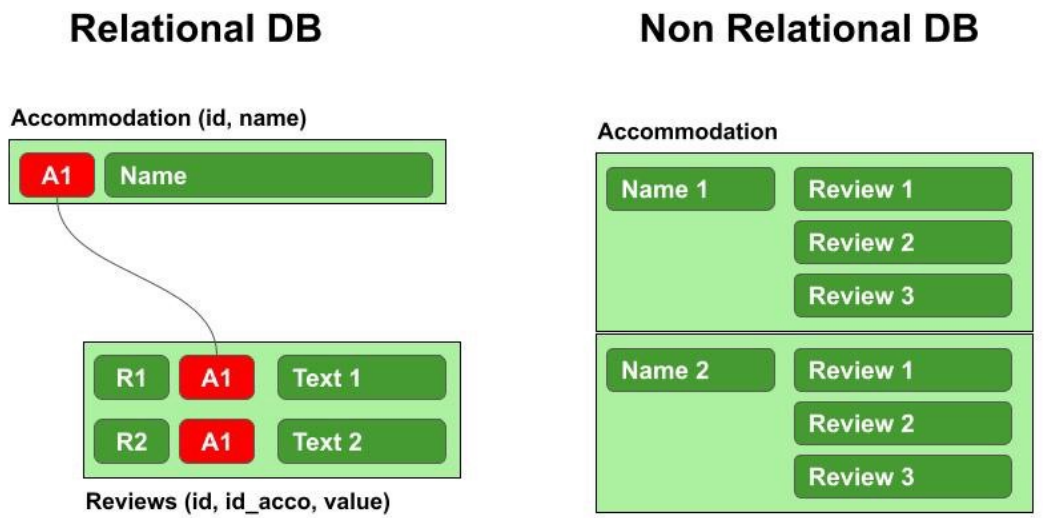
Relational Database

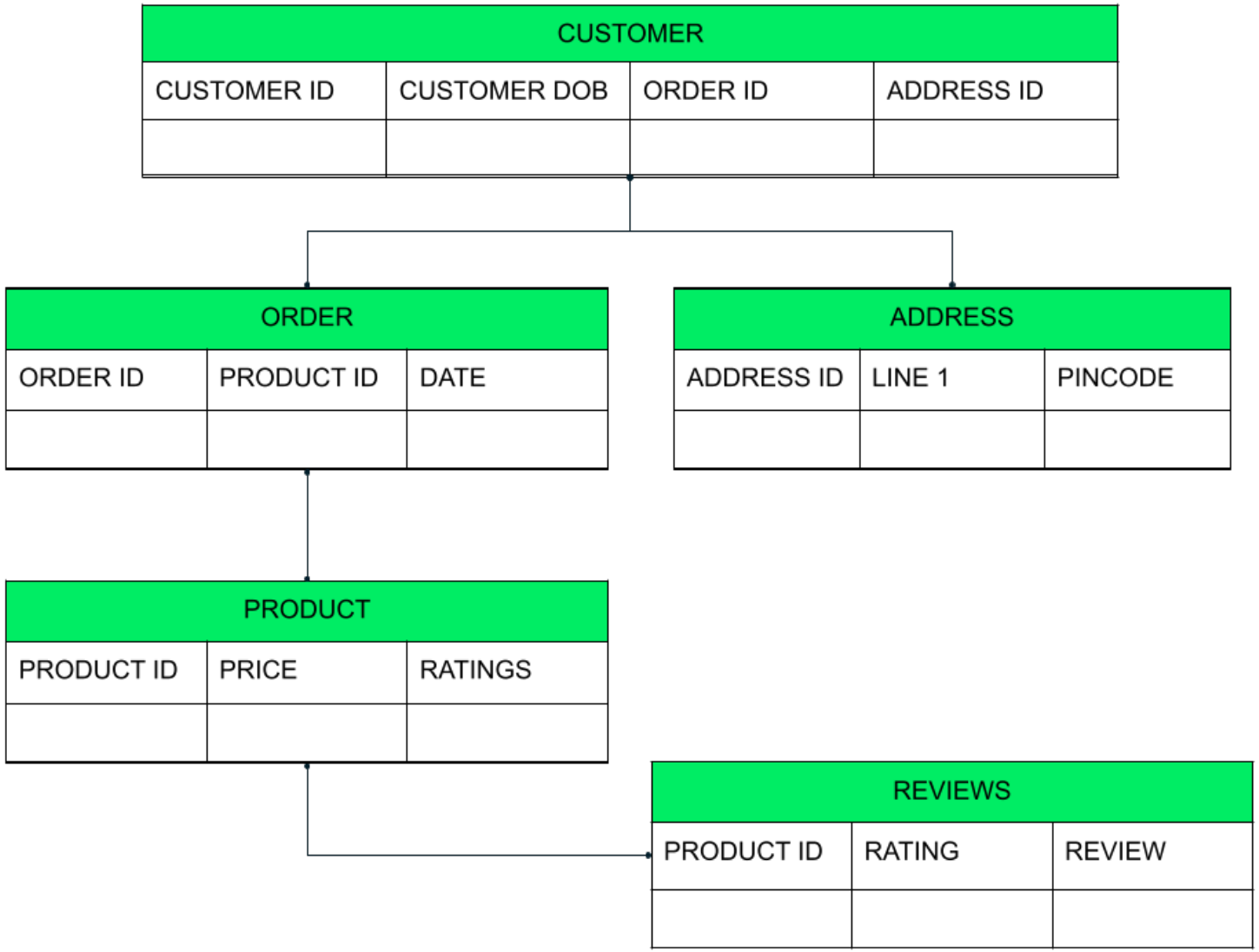
- **Normalization**: Normalization rules divide larger tables into smaller tables. Their purpose is to eliminate redundant data and ensure data is stored logically. Eg – Person has multiple address
- **Accuracy**: Tables consist of primary and foreign keys, which ensures that no duplicate data is present in data.
- **High Security**: In a relational database, we can divide the data among tables, and we can divide the tables as confidential or not. It can make our data safe.
- **Disadvantages**:
  - **Expensive**: The cost to set up a relational database is very expensive.
  - **Slow Performance**: If the database is huge and contains interconnected tables based on their primary keys. When getting the required data, the response given on the queries makes it slower, and it makes our databases quite complex to get the data.
  - **Scalability**: RDMSs are historically intended to be run on a single machine. This means that if the requirements of the machine are insufficient, due to data size or an increase in the frequency of access, you will have to improve the hardware in the machine, also known as vertical scaling. This can be incredibly expensive and has a ceiling, as eventually the costs outweigh the benefits.

Fenil Shah

Relational Database

- **Flexibility**: In relational databases, the schema is rigid. making changes to the structure of the data is very complex. You have to decide at the start what the data will look like, which isn't always possible. If you want to make changes later, you have to change all the data, which involves the database being offline temporarily.



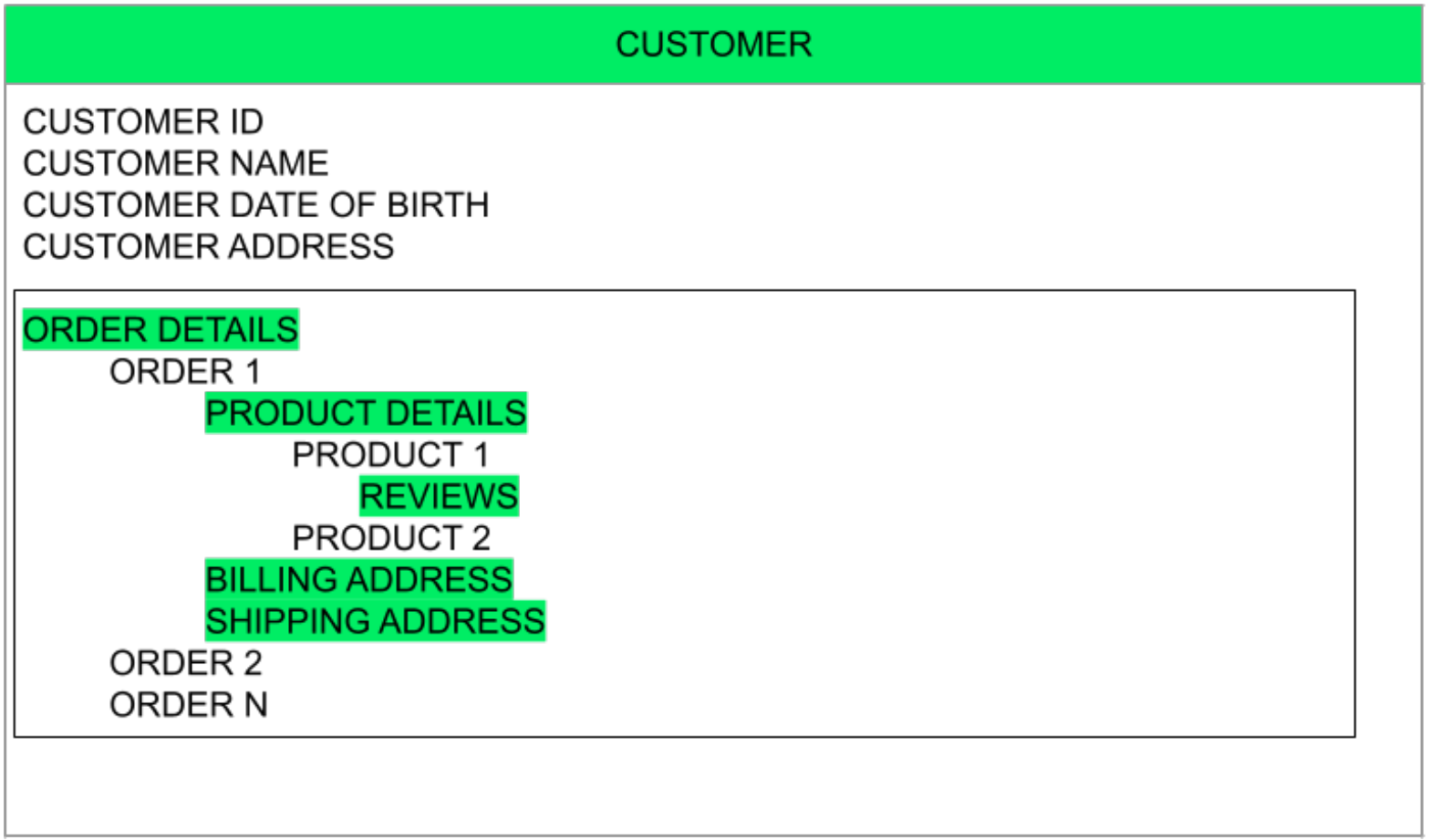

CUSTOMER
CUSTOMER ID
CUSTOMER DOB
ORDER ID
ADDRESS ID
ORDER
ORDER ID
PRODUCT ID
DATE
ADDRESS
ADDRESS ID
LINE 1
PINCODE
PRODUCT
PRODUCT ID
PRICE
RATINGS
REVIEWS
PRODUCT ID
RATING
REVIEW

Fenil Shah

Non-Relational Database

- **Performance & Fast**: These types of databases are defined for good performance, and it contains unstructured data, while in relational databases data is stored in tables, so accessing data is a bit slower than this.
- Non-relational databases are suitable for both operational and transactional data.
- They are more suitable for unstructured big data. If the data you are storing needs to be flexible in terms of shape or size, or if it needs to be open to change in future, then a non-relational database is the answer.
- Non-relational databases offer higher performance and availability. Modern NoSQL databases have been designed for the cloud, making them naturally good for horizontal scaling
- Flexible schema help non-relational databases store more data of varied types that can be changed without major schema changes.
- **Disadvantage**:
  - NoSQL doesn't guarantee ACID transactions. Disadvantage of Non-Relational Databases is that they didn't have a backup in these types of databases.
  - In NoSQL databases, there are no standardized rules of databases. The design and query language vary from one NoSQL database to another, so there is no standard process to access the data.

Fenil Shah

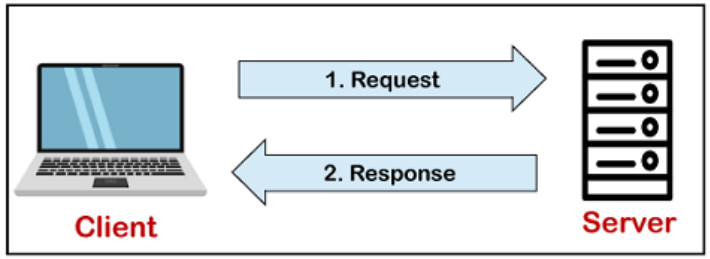

CUSTOMER
CUSTOMER ID
CUSTOMER NAME
CUSTOMER DATE OF BIRTH
CUSTOMER ADDRESS
ORDER DETAILS
ORDER 1
PRODUCT DETAILS
PRODUCT 1
REVIEWS
PRODUCT 2
BILLING ADDRESS
SHIPPING ADDRESS
ORDER 2
ORDER N

When To Use What?

- **Relational**: A project where the data is predictable, in terms of structure, size, and frequency of access. Relational databases are more suitable for data that's not likely to change frequently. You can use relational databases for medium to large datasets. if relationships between entities are important. if you have a large dataset with complex structure and relationships, embedding might not create clear enough relationships. The amount of time that RDMSs have been around also means there is wide support available, from tools to integration with data from other systems.
- **Non-Relational**: If the data you are storing needs to be flexible in terms of shape or size, or if it needs to be open to change in future, then a non-relational database is the answer. Modern NoSQL databases have been designed for the cloud, making them naturally good for horizontal scaling where a lot of smaller servers can be spun up to handle increased load. to get real-time data for operational and analytical purposes. Non-relational databases are also a good choice for AI- and IoT-based applications that need superior performance and horizontal scaling, because of the nature of big data.

MYSQL

- MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database.
- MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services.
- The core of the MySQL database is the MySQL Server. This server is available as a separate program and responsible for handling all the database instructions, statements, or commands. The working of MySQL database with MySQL Server are as follows:
- MySQL creates a database that allows you to build many tables to store and manipulate data and defining the relationship between each table.
- Clients make requests through the GUI screen or command prompt by using specific SQL expressions on MySQL.
- Finally, the server application will respond with the requested expressions and produce the desired result on the client-side.



MYSQL

- Easy to use. It is secure. Client/ Server Architecture. Free to download. It is scalable. Speed. Compatible on many operating systems. Allows roll-back. Memory efficiency. High Performance. High Productivity. Partitioning.
- CREATE DATABASE bookdb; SHOW DATABASES; USE bookdb; DROP DATABASE bookdb;
- CREATE TABLE book(id int AUTO_INCREMENT, name varchar(45) NOT NULL, cost int NOT NULL, PRIMARY KEY (id));
- SHOW TABLES; DESCRIBE book; ALTER TABLE book ADD author varchar(40) NOT NULL;
- SELECT * FROM book; DROP TABLE book;
- INSERT INTO People (id, name, cost, author) VALUES (1, 'Peter', 100, 'R.D. Sharma');
- UPDATE book SET author = 'Stephen Hawking' WHERE id = 2;
- DELETE FROM book WHERE id=2;
- SELECT author. fname, author. lname, book.name FROM author INNER JOIN book ON author.author_id = technologies.tech_id;

Fenil Shah