(BCS-502)

UNIT-04 (LEC-01)

- Javabeans
- Javabeans Properties [AKTU-21-22, 22-23]
- Types of beans
- Stateful Session bean [AKTU-22-23]
- Stateless Session bean [AKTU-22-23]
- Entity bean

Page No.:
Date: / /

Javabean :

- According to Java white paper, it is a reusable software component. A bean encapsulates many objects in one object so that we can access this object from multiple places. Moreover, it provides easy maintenance.
- Reusability is the main concept in any Programming language.
- Javabeans is a Portable, platform-independent model written in Java Programming Language.
- Javabeans contain several elements like constructors, Getter / setter methods

Preparing a class to be a Javabeans :-

- It should have a no-arg constructor - default constructor
- It should be Serializable. Beans should implement java.io.serializable as it allow to save, store and restore the state of a Javabean you are working on.

Page No.:
Date / /

- It Should Provide methods to set and get the values of the Properties known as getter and Setter method.
- A getter method used to read the value of a readable property. To update the value, a setter method should be called.

Example:-

```
public class TestBean{
private String name;
public void SetName (String name)
{
 this.name = name;
}
public String getName()
{
 return name;
}
}
```

Setter Method :-

- It Should be Public in nature.
- The return-type should be Void.
- It should take Some argument.

Getter Method :-

- It Should be Public in nature.

- The return-type should not be Void.
- It should not take any argument.

Creating a JavaBeans :

→ Implements java.io.Serializable
→ Declare Private Variable
→ Declare no-arguments Constructor
→ Declare getter and setter

JavaBeans Properties : [AKTU-2021-22, 2022-23]

1. getPropertyName().
→ Suppose the Property name is FullName; you will be required to use getFullName() as the method name to read the full name of a Person.
→ It doesn't take any argument
→ It is Public in nature
→ It is Prefixed with the term 'get'
→ It don't have a void return type. It should have a return type.

2. setPropertyName()
Suppose the Property name is FullName, setFullName() is the method name to use to write the FullName.
mutator is the name of this method mentioned below Properties:-
→ It takes some argument

→ It is Public in nature

→ It is Prefixed with the term 'set'

→ It has a void return type

Properties can be of types:

1. Read only Property:

→ It has a getter method only

Ex:-
```
public class TestBean {
private String name;
//getter method
public String getName()
{
return name; //return
}
}
```

2. Write only Property:

→ Contain only setter method.

Ex:-
```
public class TestBean {
private String name;
//setter method
public void setName(String name) {   // argument
this.name = name;
}
}
```

3. Read and Write Property:

→ Contain setter and getter method both

Ex:- public class TestBean {
private String name;
// setter method

public void SetName (String name) {
this.name = name;
}
// getter method
public String getName()
{
return name;
}}

4) Indexed Properties:-
- An Index Property is an array instead of a single value. In this case, the bean class provide a method for getting and setting the entire array.

Advantages and Disadvantage of Javabean

- The properties, events and method of a bean can be exposed to another application.
- A bean may register to receive events from other Objects

Page No..
Date / /

- Javabeans are inherently mutable and so lack the advantages offered by immutable objects.

Enterprise JavaBeans:

- Enterprise Java beans is one of the several Java API for standard manufacture of enterprise software.
- EJB is a server-side software element that summarize business logic of an application.
- Such machine code addresses the same types of problems, and solution to these problem are often repeatedly re-implemented by programmers.

Types of Javabeans [AKTU-2022-23]

1. Session Bean:-
Session bean contain business logic that can be invoked by local, remote or webservice client.

Two-type of Session beans:
1. Stateful Session bean
2. Stateless Session bean

(i) Stateful Session bean :-
Stateful Session bean performs business task with the help of a state. Stateful Session bean can be used to access various method calls by storing the information in an instance variable

(ii) Stateless Session bean :-
Stateless Session bean implement business logic without having a persistent storage mechanism

2 Message Driven Bean :-
Like Session Bean, it contain the business logic but it invoked by passing message.

3 Entity Bean :-
It summarizes the state that can be remained in the database. Now, it is replaced with JPA (Java persistent API)

Two types of Entity Bean :-

(i) Bean managed persistent :-
In a bean managed persistent type of entity bean, the programmer has to write the code for database calls.

(iii) Container Managed Persistence

Container managed persistence are enterprise bean Persists across database

Date : __/__/____

UNIT-04 (LEC-2)

- Introduction
- Environment Setup
- REPL Terminal
- Server
- NPM
- API and Endpoints
- Callbacks Concept
- Events
- Packaging
- Express Framework

SUBSCRIBE

JOIN TELEGRAM

Date : __/__/____

Introduction of Node.js :-

- Initially, JS was used in Frontend only, to enhance UI and interaction on website
- Node.js allow developer to use same language in Frontend and Backend i.e Client side as well as Serverside
- Node.js was developed by Ryan Dhal
- Node.js is a javascript runtime build on Chrome V8 JS Engine.

V8 Engine

- developed by Google ECMAScript

Human readable Code → (V8 Engine) → Machine Readable Code

- Node.js is a cross-platform, Open-Source maintained by (Open JS Foundation). JS runtime environment that can run on "Window", "Linux" or more
- Node.js help us to run JS outside of the browser.

Date : __/__/____

History Of Node.js

1. **2009** :-
   - → Ryan Dhal Started Node.js
   - → Initially Started on Firefox
   - → Ryan Dhal developed non-blocking server which helps in multiple req with threads.

2. **2010** :-
   - → NPM was Introduced

3. **2011**
   - → Window Support For Node.js

4. **2012**
   - → Ryan left the node JS team
   - → Issac become the new leader

5. **2014**
   - → becoz of the slow Pace of development in Node.js

6. **2015**
   - → Fedor and Joyent merged and Node.js Foundation.

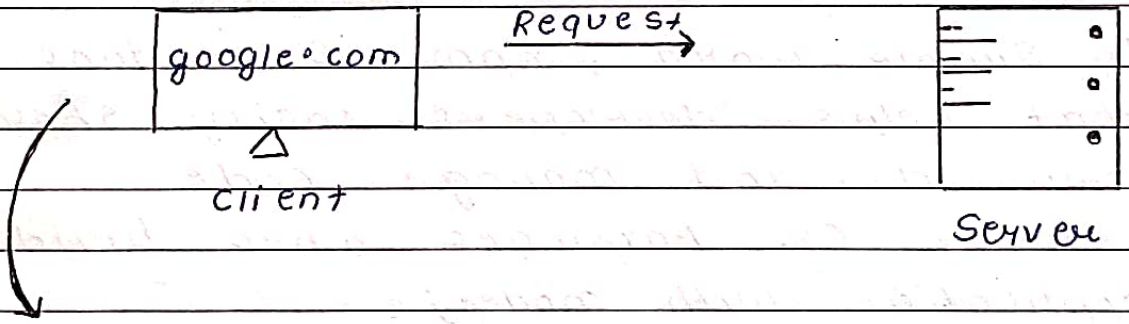
7. **2019**
   - → JS Foundation & Node JS Found°

Date : __/__/____

merged and Open JS Foundation

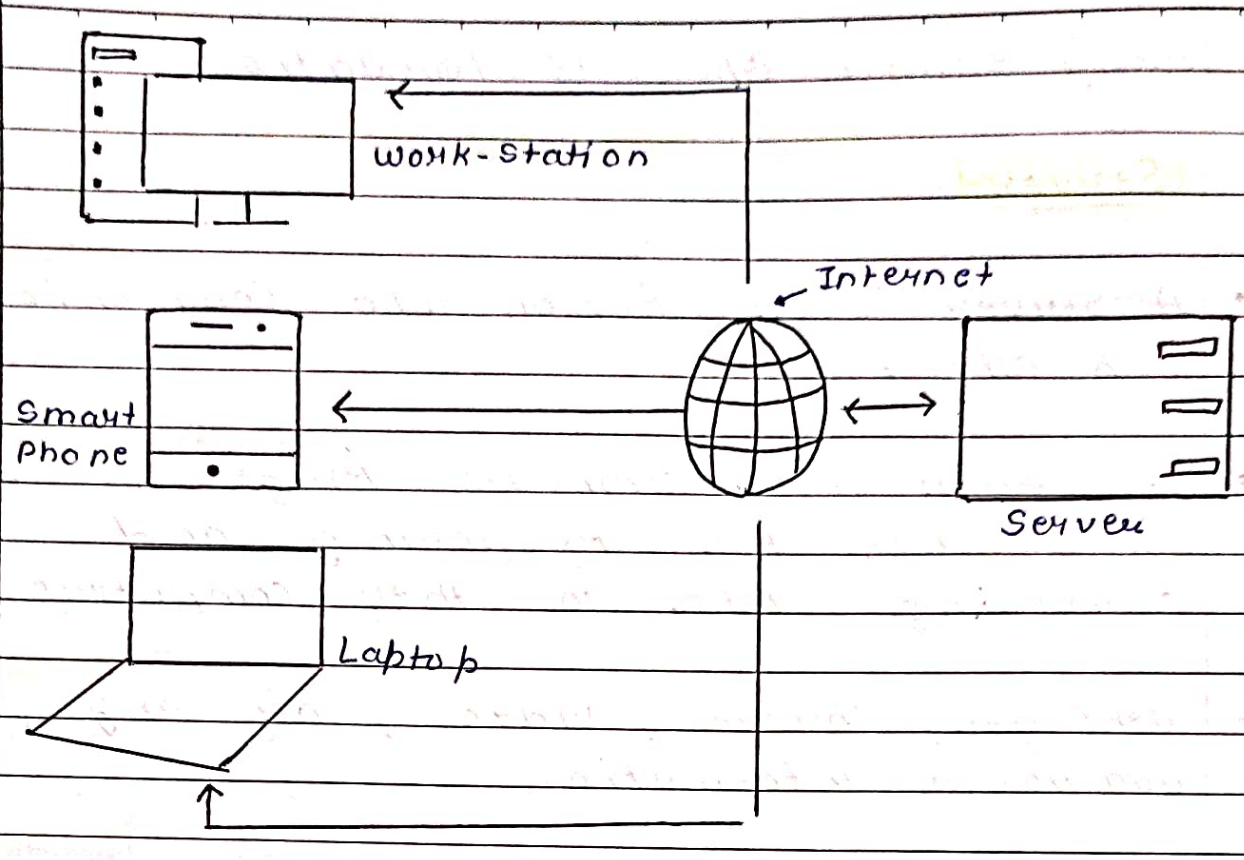
Server :-

- A Server is a Person who Communicate with client
- A server is a computer Program that's responsible for Preparing and delivering data to other computers.
- WebPages, images, videos, or any additional information
- Creating a Server in Node.js via express Package



Initially JS was the main language which was used inside the browser	→	But after Node.js we can run Javascript on the server.

Date : __/__/____



NPM (Node Package Manager)

- In simple word, npm is a tool that helps developers easily share, download, and manage code libraries or packages when building application with node.js

- Imagine npm as a huge library of pre-built code-pieces - each solving a different problem. Instead of writing every piece of code from scratch, you can use npm to find and install packages created by others.

Date :__/__/____

- In this time NPM is a platform same as Playstore and AppStore
- At this time we can upload the code in form of package in NPM platform
- Right now, npm contain over 800,000 packages for various Opp^n from front-end, robotics.

Note :- CPU intensive work cannot done by node.js's like Image Processing.

~~Package~~

Client —Get/Post→ Server → DB

HTTP methods :-

- Get :- To receive data.
- Post :- To send or submit data to the server, usually to create something new.
- Patch :- To Partially Update data on the server.
- Put :- To Update existing data on the server
- Delete :- To remove data from the server.

Date : __/__/____

JSON :-

- JSON is a bit like the organized Format for exchanging data b/w Computer.
- JSON is a lightweight
- Structure & Organized data
- JSON is represented as a string " "
- Imagine you're sending a msg to your Friend, and you want to include information like - name, age, list and hobbies
- You need to Organize the infor^m in a way that both you and your Friend Understand

Package.json :- To ensure a list of Packages with their Version.

Package-lock.json :- ensure detailed of every package installed with Version, sub dependencies store detailed, discount everything.

Date : __/__/____

- It's like a detailed billed

What are API and Endpoints :-

- An API is like a bridge that allows two different software application to talk to each other.

- API are used to access data or function-ality on your server. When a client makes a request to your API, it sends data and asks the server to return certain information.

For-ex :- Collection of list = menu card = API

Endpoints :- Endpoint is a specific URL on your server that represents one operation or data access point in your API.

For-ex :- Option in that list = = Endpoints.

Callback Concept :-

- The Callback Concept is crucial for handling tasks that take time to complete, like reading a file, querying a database, or making an API request.

Date : __/__/____

- A Callback is simply a function that is passed as an argument to another function.

- Node.js is asynchronous and non-blocking by default, meaning it can handle multiple tasks at the same time.

For-ex :-

```
Const fs = require ('fs');   ← file-system

fs.readFile ('example.txt', 'utf8', (err, data)
    => {
    if (err) {
        Console.log ('Error reading :', err);
        return;
    }
    Console.log ('file content:', data);
});
```

Summarizing the Flow :-

Step-1 Import the fs module.

Step 2 Call fs.readFile to read example.txt. Pass a callback fun^ to handle the result

Date : __/__/____

Step 3 Inside the callback, check if an error occured

→ if there an error, print it & exit.

→ if no error, log the file content (data).

Events in Node.js :-

- Every action on a computer is an event. like when a connection is made or a file is opened.

- Node.js is built on an event-driven architecture, meaning it uses events to manage and coordinate asynchronously

key Concepts Of Events in Node.js

1. **Event Emitter** :- is a core feature in Node.js that lets you set up communication system within your application

→ Event.

→ Listener

```
Const EventEmitter = require('events');
Const myEmitter = new EventEmitter();
```

Date : __/__/____

Express Frameworks :

- Express is a popular web application framework for Node.js, designed to make it easier to build web applications and APIs

- Think of Express as a helpful toolkit that simplifies many of the complex tasks involvind in building web servers, handling request and managing responses.

Why Use Express ?

- Node.js on its own can handle HTTP request, but it requires a lot of code to handle even simple tasks

- Express simplifies this process by providing a structure and helpful tools so you can create web servers quickly and easily

Key Features Of Express

1. Routing
2. Middleware
3. Easy integration with DB
4. Error Handling

Web-Technology

Date : __ / __ / ____

UNIT-04 (LEC-3)

- Database
- MongoDB
- Create Database
- Create Collection
- Operations.

SUBSCRIBE

Date : __/__/____

Database :-

- A database is a structured collection of data that is stored electronically and organized for easy access, management, and updating.

- Ultimately Let's suppose we are going to open Restaurant & there is lots of data around it
like → no of chefs, menu details

- A database in MongoDB is a collection of documents that can be stored in the cloud or locally on a machine

Lots of Database are available :-

→ SQL → Maria DB
→ PostgreSQL → Oracle
→ MongoDB

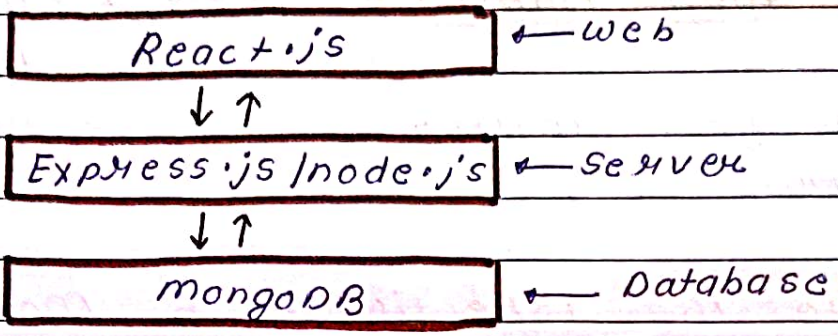
- Database typically have their own server system to manage & provide access to the data they store

- These database server system are seprate from node.js's server but work together to create dynamic & data-driven web application

Date : __/__/____

MongoDB :-

- MongoDB is a type of database that stores information in a way that's a bit like how you'd keep notes in a notebook.



Node.js Server and Database Server :-

- Database Server store your application data. When your node.js Server needs data, it sends request to the DB server, which then retrieves and sends the requested data back to the node.js Server.

SQL Syntax	Mongodb Syntax
Table	Collection
Row / Record	Document
Column	Field

Date : __/__/____

Create Collection in Database :-

1. **Open MongoDB :-** Start by Opening management tool like MongoDB Compass.

2. **Choose Your Database :** Select or Create the database where you want your new Collection to be.

3. **Create the Collection :-** In MongoDB Collections are usually created automatically when you insert your first document.

```
db.createCollection("user"); ← Create Table
```

4. **Check Your Collection (optional) :-** To see if your Collection was Created, you can list all Collections in the DB by typing

```
show collections.
```

Operations in Mongodb :-

Date : __/__/____

1. **Insert Data into a Collection**:- To add a new document into a collection, use the insertOne or insertMany method.

```
// Insert one document

db.CollectionName.insertOne({name:"Alice", age:25})

// Insert multiple document

db.CollectionName.insertMany([
  {name:"Bob", age:30},
  {name:"Charlie", age:35}
])
```

2. **Delete Data From Collection**:- To remove document from a collection, you can use deleteOne or deleteMany

```
// Delete one document
db.CollectionName.deleteOne({name:"Alice"})

// Delete multiple document
db.CollectionName.deleteMany({age:{$gt:30}})
```

Date : __/__/____

3. **Update Data in Collection** :- To modify data, use updateOne or updateMany

```
//Update one document
db.CollectionName.updateOne(
    {name: "Bob"},
    {$set: {age: 32}}
)

//Update multiple documents.
db.CollectionName.updateMany(
    {age: {$lt: 30}},
    {$set: {status: "Young Adult"}}
)
```

4. **Query Data in a Collection** :- To retrieve documents use Find.

```
// Find all users with age greater than 25

db.CollectionName.find({age: {$gt: 25}})
```

5. **Sort Data in a Collection** :-

To sort result, use sort.

Date : __/__/____

```
// Find all user and sort by age in
descending order

db.CollectionName.find().sort({age:-1})
```

6. Join Data From Different Collection :-
In mongodb, joins are done using the aggregate function and the $lookup stage. This lets you combine data from two collection based on common field.

```
// Assume we have two collection: users & orders

db.users.aggregate([
  {
    $lookup:{
      from: "orders",
      localField: "_id",
      foreignField: "user_id",
      as: "user_orders"
    }
  }
])
```

- Now, currently we don't have a such frontend thing, so we are using Postman for this.

Date : __/__/____

Connect mongodb with Node.js :-

- Now, to connect mongodb with node.js we need a mongodb driver. The driver acts as a bridge b/w node.js aplⁿ and the mongodb database
- The driver translate the JS code into a format the mongodb can understand & vice versa.