DISTRIBUTED SYSTEMS

- Distributed systems are computer systems made up of multiple independent components that work together to accomplish a common goal.

- These components are physically separate and can be located in different geographic locations, but they communicate and coordinate with each other over a network.

- The goal of a distributed system is to improve performance, reliability, and scalability by breaking down a large task into smaller sub-tasks that can be executed in parallel on different components of the system.

- Examples of distributed systems include cloud computing platforms, peer-to-peer networks, and distributed databases.

Latency and throughput are two important metrics in software engineering that describe different aspects of system performance.

Latency measures the speed of a single request, throughput measures the system's overall capacity for handling a high volume of requests.

Distributed systems software infrastructure refers to the various components and technologies used to build and manage large-scale distributed systems.

The infrastructure for these systems is built around three key components: storage, communication, and computation.

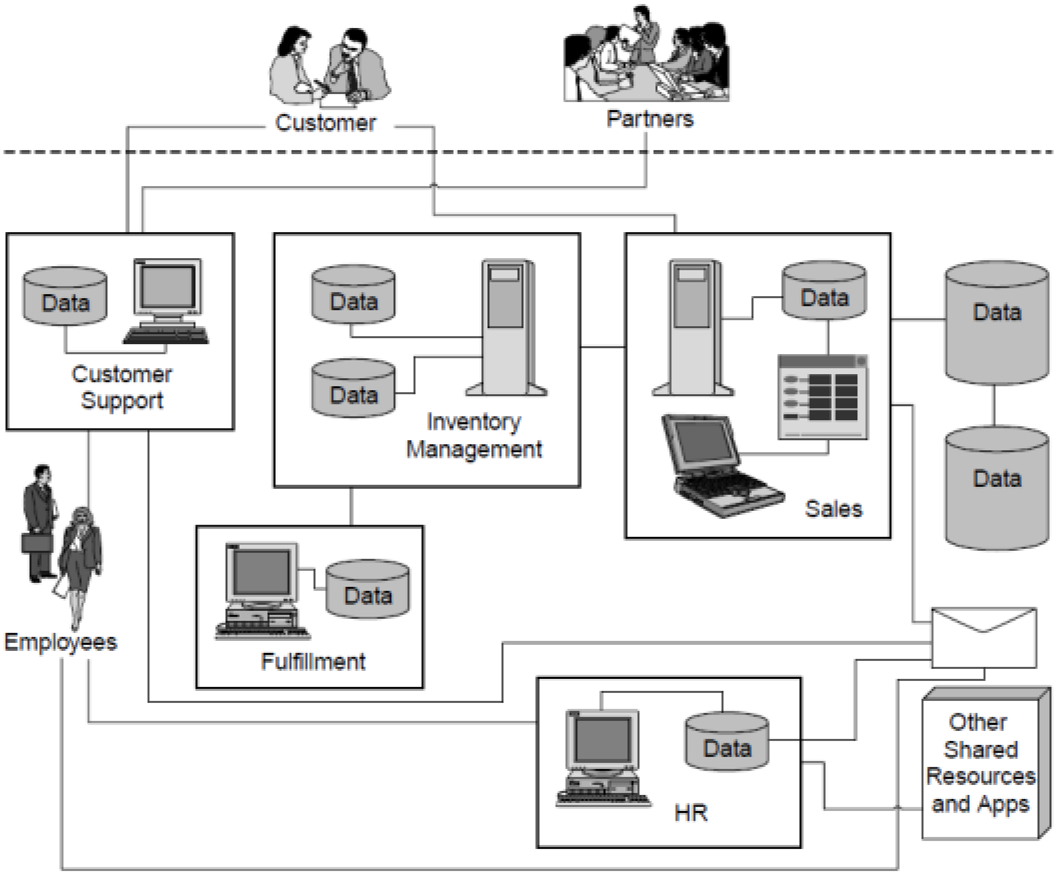
Storage refers to the way data is stored and managed across the distributed system.

Communication is the way that nodes in the distributed system exchange information and coordinate their actions.

Computation refers to the way that processing power is distributed across the system.

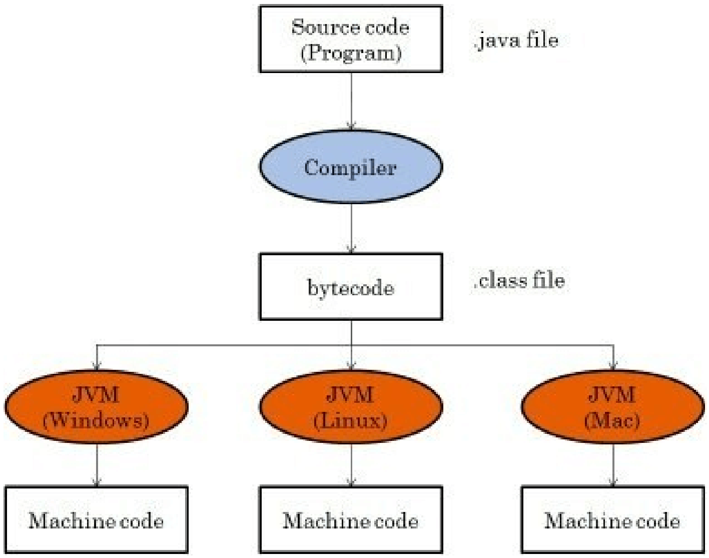
ENTERPRISE SOFTWARE

Enterprise software is computer software that assists with the needs of an organization rather than individual users. it's a large-scale software designed to be used in a corporate environment that caters to many users and user roles.



WHY YOU SHOULD CHOOSE JAVA FOR ENTERPRISE APPLICATIONS

Cross-Platform Compatibility - It means that apps written in Java can run seamlessly on almost all existing platforms, consisting of Windows, Mac OS, and Linux, which makes it more apt for enterprises.



- Simple to Use
- Stable Language
- Speed and performance
- Powerful Development Tool
- Availability of Libraries
- Security
- Rich API

WHERE IS IT USED?

Java finds its implementation in almost every industry and domain presented on the market.

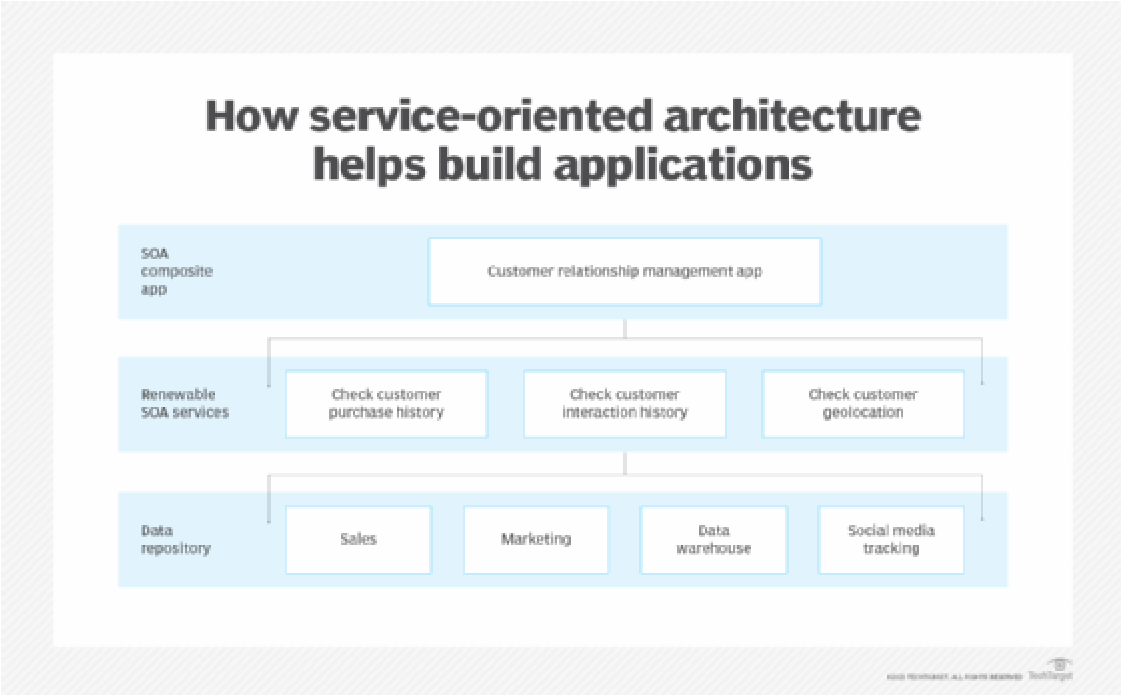
- Healthcare (EHR, PMS, remote patient monitoring)
- Finance (online banking, digital wallets, trading platforms)
- eCommerce (eCommerce platforms, marketplaces, systems of sales analytics)
- eLearning (LMS and distant learning platforms)
- Entertainment and Media (live streaming, networking apps, content delivery)
- Manufacturing (ERP, data processing, IoT) and many more

SERVICE ORIENTED ARCHITECTURE

Service-oriented architecture (SOA) is a method of software development that uses software components called services to create business applications. Each service provides a business capability, and services can also communicate with each other across platforms and languages.

Some SOA product has been built by Oracle (SOA Suite), IBM(Websphere), Microsoft(BizTalk)

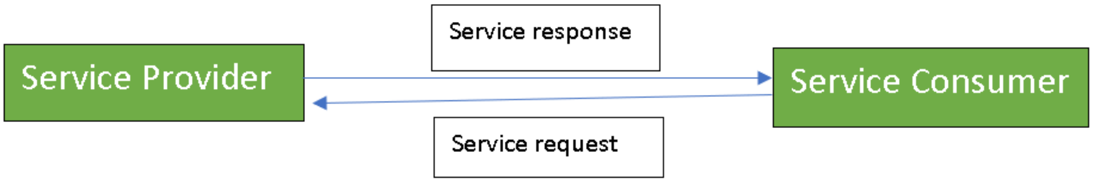
SOA simplifies complex software systems into reusable services that can be accessed by other applications and users referred to as service consumers. These services can be used as building blocks of new applications.


How service-oriented architecture helps build applications
SOA composite app
Customer relationship management app
Renewable SOA services
Check customer purchase history
Check customer interaction history
Check customer geolocation
Data repository
Sales
Marketing
Data warehouse
Social media tracking
TechTarget

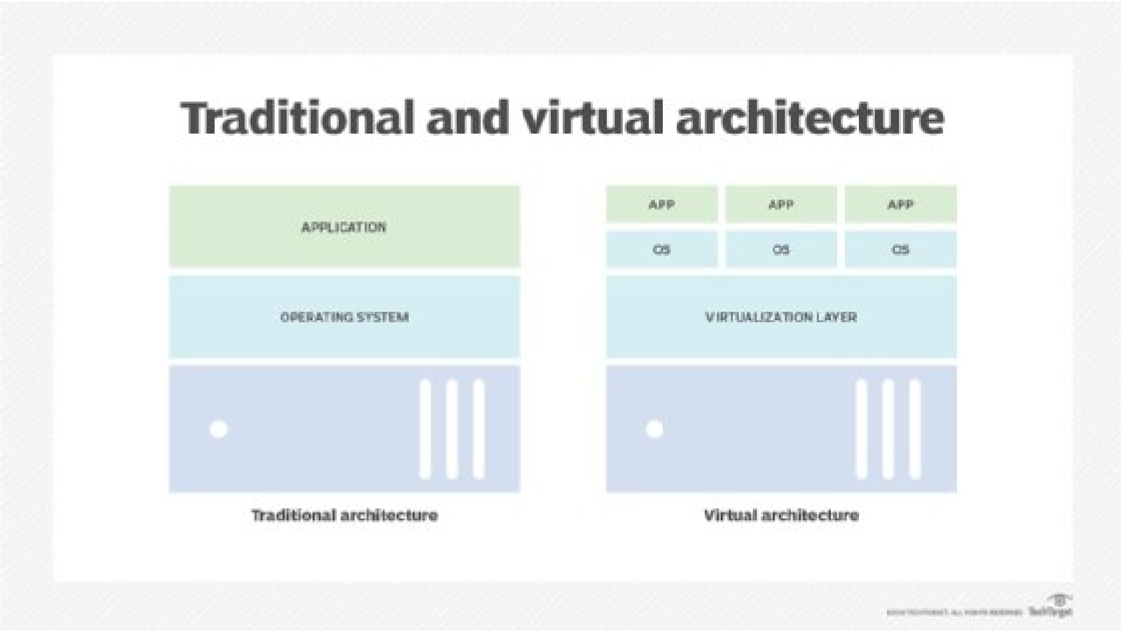
THERE ARE TWO MAJOR ROLES WITHIN SERVICE-ORIENTED ARCHITECTURE

Service provider. This component creates or provides the service. Most organizations either create their own service or use third-party services.

Service consumer. The service consumer is the individual, system, application or other service that makes service requests to the service provider.

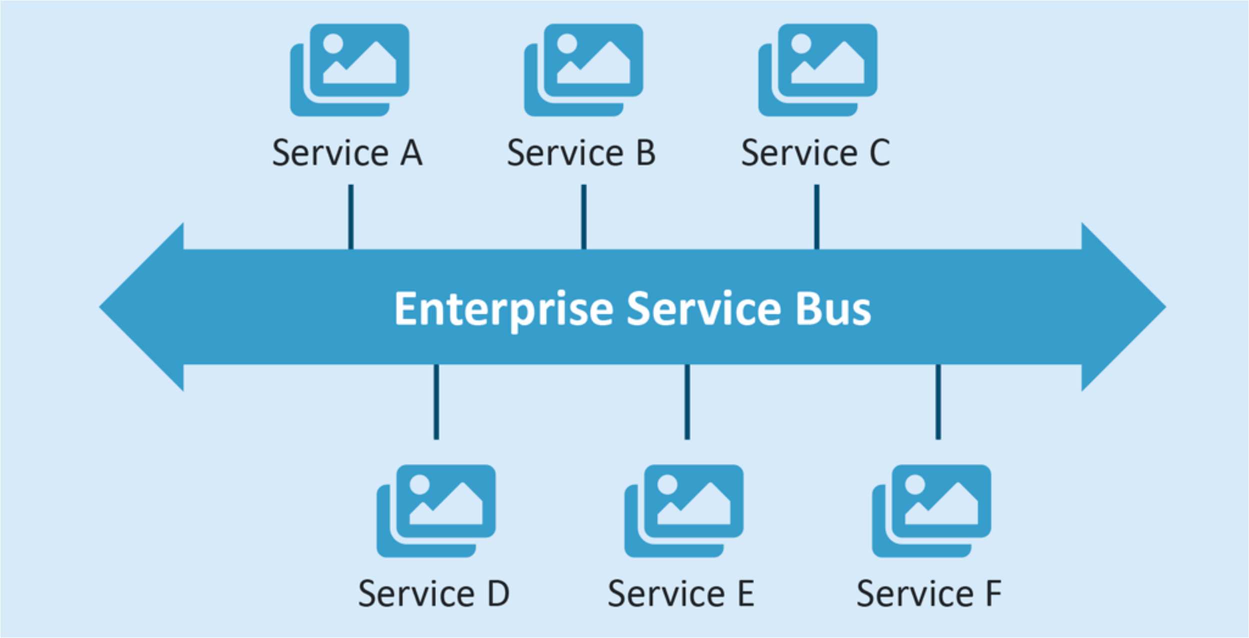


While part of traditional architecture, SOAs introduced principles used by virtual architectures as well.



ENTERPRISE SERVICE BUS

The idea of an ESB is that many components can connect in a standardized way and then communicate over the bus with any other component.



GUIDING PRINCIPLES OF SOA

Standardized service contract: Specified through one or more service description documents.

Loose coupling: Services are designed as self-contained components, maintain relationships that minimize dependencies on other services.

Abstraction: A service is completely defined by service contracts and description documents. They hide their logic, which is encapsulated within their implementation.

Reusability: Designed as components, services can be reused more effectively, thus reducing development time and the associated costs.

Autonomy: Services have control over the logic they encapsulate and, from a service consumer point of view, there is no need to know about their implementation.

BENEFITS/ ADVANTAGES OF SERVICE-ORIENTED ARCHITECTURE

Service reusability: In SOA, applications are made from existing services. Thus, services can be reused to make many applications.

Easy maintenance: As services are independent of each other they can be updated and modified easily without affecting other services.

Platform independent: SOA allows making a complex application by combining services picked from different sources, independent of the platform.

Availability: SOA facilities are easily available to anyone on request.

Reliability: SOA applications are more reliable because it is easy to debug small services rather than huge codes

Scalability: Services can run on different servers within an environment, this increases scalability

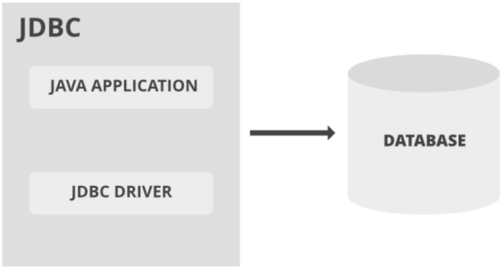
DISADVANTAGES OF SOA:

- **High investment:** A huge initial investment is required for SOA.
- **Complex service management:** When services interact they exchange messages to tasks. the number of messages may go in millions. It becomes a cumbersome task to handle a large number of messages.
- **Increase the Network Traffic**

What is JDBC?

Java™ database connectivity (JDBC) is the JavaSoft specification of a standard application programming interface (API) that allows Java programs to access database management systems. The JDBC API consists of a set of interfaces and classes written in the Java programming language.

Using these standard interfaces and classes, programmers can write applications that connect to databases, send queries written in structured query language (SQL), and process the results.



We can use JDBC API to handle database using Java program and can perform the following activities:

- Connect to the database
- Execute queries and update statements to the database
- Retrieve the result received from the database.

The java.sql package contains classes and interfaces for JDBC API. A list of popular interfaces of JDBC API are given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface

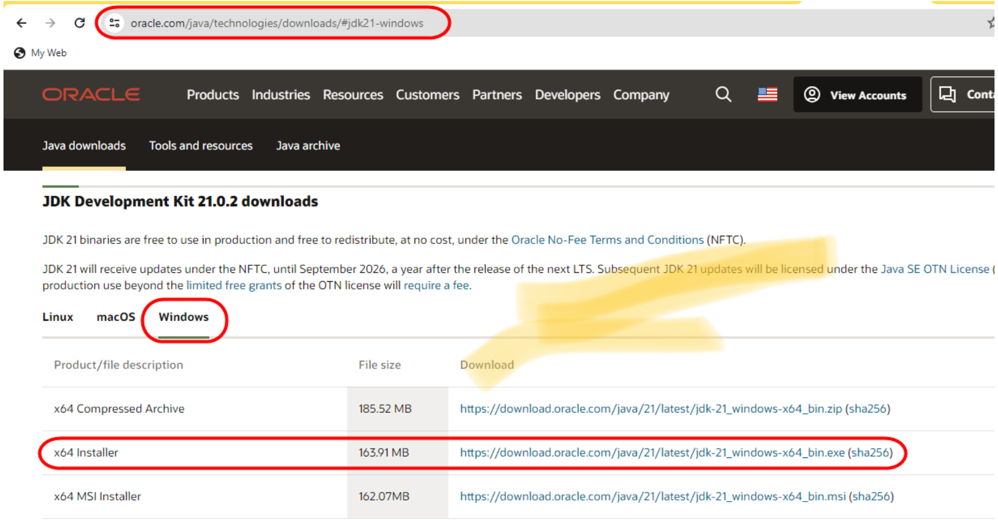
There are 4 steps to connect any java application with the database using JDBC. These steps are as follows:

- Create connection
- Create statement
- Execute queries
- Close connection

ESTABLISH CONNECTION WITH THE MYSQL DATABASE

- **Install Java JDK**

https://www.oracle.com/java/technologies/downloads/#jdk21-windows



JDK Development Kit 21.0.2 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the Java SE OTN License (
production use beyond the limited free grants of the OTN license will require a fee.

Product/file description	File size	Download
x64 Compressed Archive	185.52 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256)
x64 Installer	163.91 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer	162.07MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)

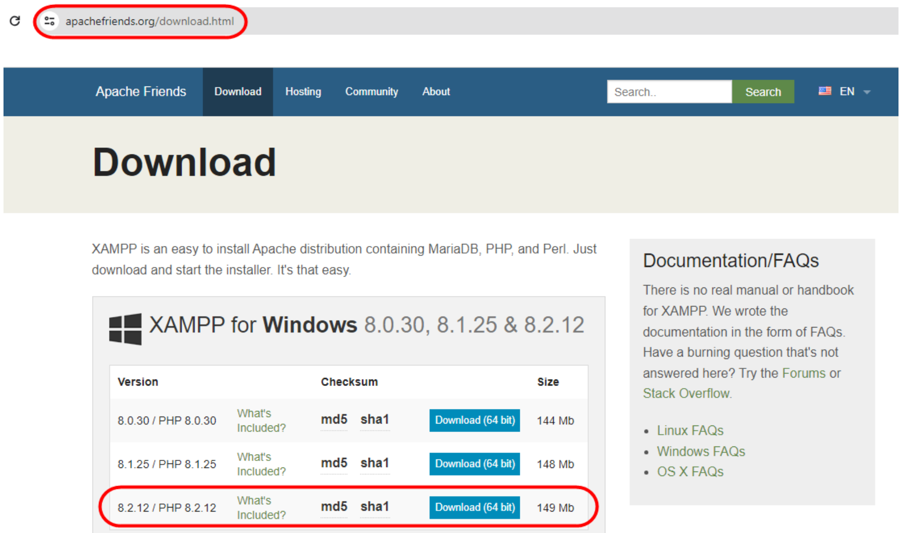
- **Install MySql Database Server**

XAMPP Installation

https://www.apachefriends.org/download.html

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

It included phpMyAdmin and which is a free software tool written in PHP, intended to handle the administration of MySQL/MariaDB over the Web.



Download

XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

XAMPP for **Windows** 8.0.30, 8.1.25 & 8.2.12

Version		Checksum			Size
8.0.30 / PHP 8.0.30	What's Included?	md5 sha1		Download (64 bit)	144 Mb
8.1.25 / PHP 8.1.25	What's Included?	md5 sha1		Download (64 bit)	148 Mb
8.2.12 / PHP 8.2.12	What's Included?	md5 sha1		Download (64 bit)	149 Mb

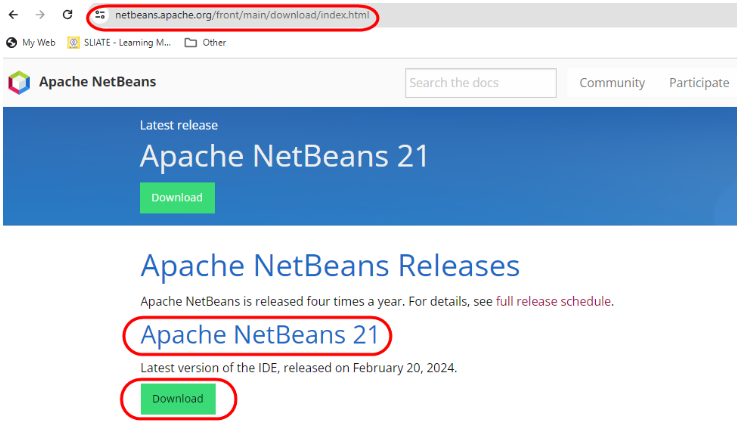
Documentation/FAQs

There is no real manual or handbook for XAMPP. We wrote the documentation in the form of FAQs. Have a burning question that's not answered here? Try the Forums or Stack Overflow.

- Linux FAQs
- Windows FAQs
- OS X FAQs

- **Install NetBeans IDE**

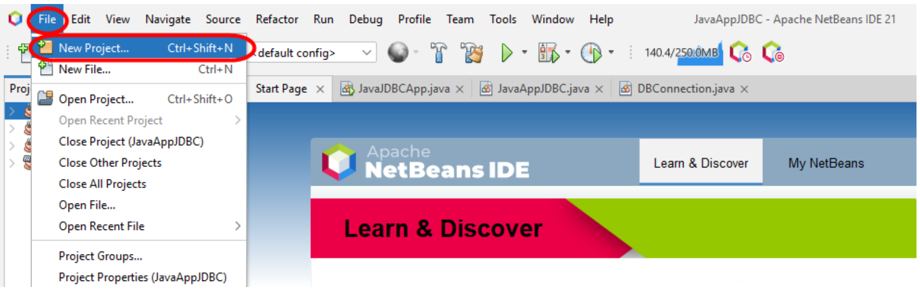
NetBeans is an integrated development environment (IDE) for Java and it provides editors, wizards, and templates to help you create applications in Java

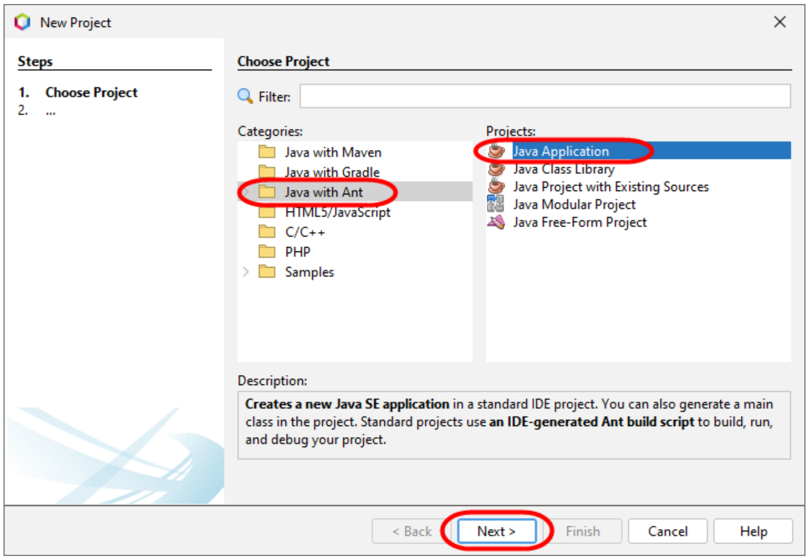


Apache NetBeans is released four times a year. For details, see full release schedule.

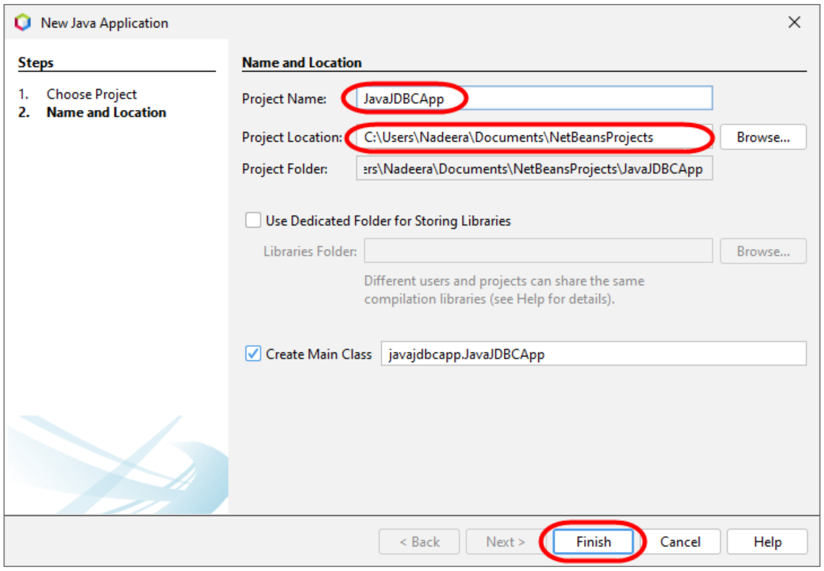
Latest version of the IDE, released on February 20, 2024.

01. Open NetBeans IDE and Create a New Project





Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: JavaJDBCApp

Project Location: C:\Users\Nadeera\Documents\NetBeansProjects

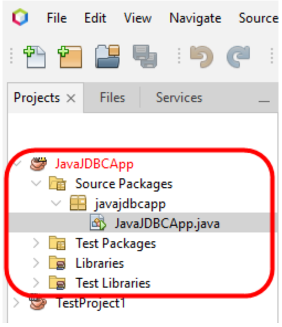
Project Folder: ers\Nadeera\Documents\NetBeansProjects\JavaJDBCApp

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

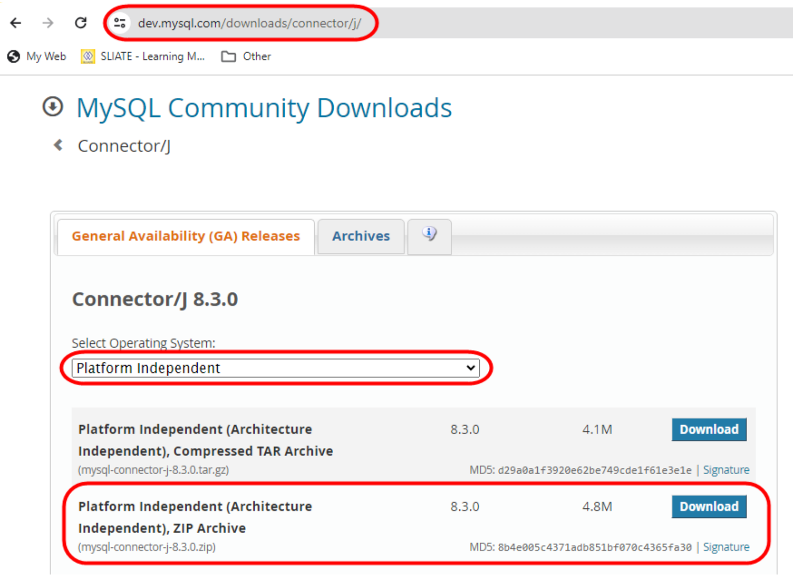
☑ Create Main Class javajdbcapp.JavaJDBCApp



```
JavaJDBCApp
  Source Packages
    javajdbcapp
      JavaJDBCApp.java
  Test Packages
  Libraries
  Test Libraries
TestProject1
```

02. Download MySQL JDBC Driver (Connector/J 8.3.0)

MySQL Connector/J is the JDBC driver for MySQL and MariaDB.



dev.mysql.com/downloads/connector/j/

MySQL Community Downloads

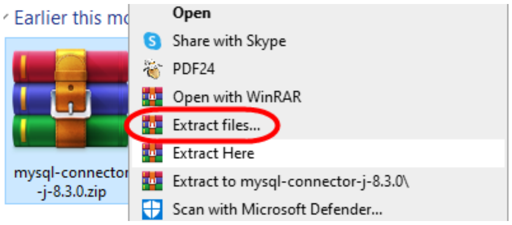
Connector/J

Connector/J 8.3.0

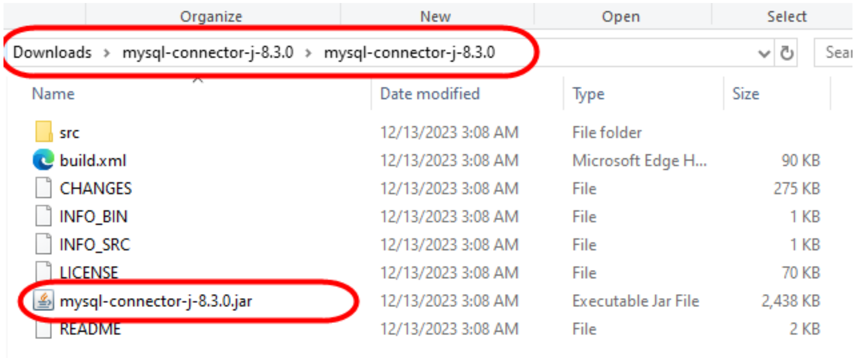
Select Operating System: Platform Independent

Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-j-8.3.0.tar.gz)	8.3.0	4.1M	Download MD5: d29a0a1f3920e62be749cde1f61e3e1e | Signature
Platform Independent (Architecture Independent), ZIP Archive (mysql-connector-j-8.3.0.zip)	8.3.0	4.8M	Download MD5: 8b4e005c4371adb851bf070c4365fa30 | Signature

03. Extract downloaded mysql-connector-j-8.3.0 Zip file


Earlier this mo
mysql-connector-j-8.3.0.zip
Open
Share with Skype
PDF24
Open with WinRAR
Extract files...
Extract Here
Extract to mysql-connector-j-8.3.0\
Scan with Microsoft Defender...

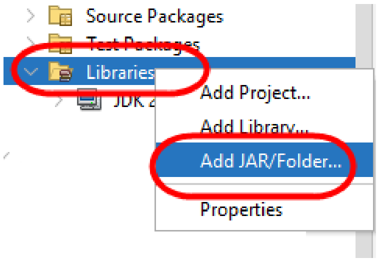
A jar file is a collection of Java classes (usually a library).

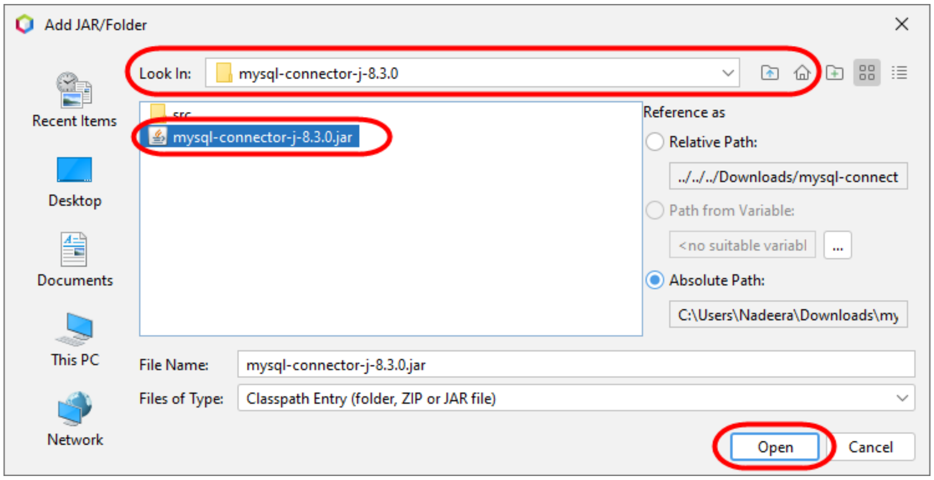

Organize
New
Open
Select
Downloads › mysql-connector-j-8.3.0 › mysql-connector-j-8.3.0
Sear

Name	Date modified	Type	Size
src	12/13/2023 3:08 AM	File folder	
build.xml	12/13/2023 3:08 AM	Microsoft Edge H...	90 KB
CHANGES	12/13/2023 3:08 AM	File	275 KB
INFO_BIN	12/13/2023 3:08 AM	File	1 KB
INFO_SRC	12/13/2023 3:08 AM	File	1 KB
LICENSE	12/13/2023 3:08 AM	File	70 KB
mysql-connector-j-8.3.0.jar	12/13/2023 3:08 AM	Executable Jar File	2,438 KB
README	12/13/2023 3:08 AM	File	2 KB

04. Add mysql-connector-j-8.3.0.jar file to the Project

To connect java application with the mysql database, mysql-connector-j.jar file is required to be loaded.





Look In: mysql-connector-j-8.3.0

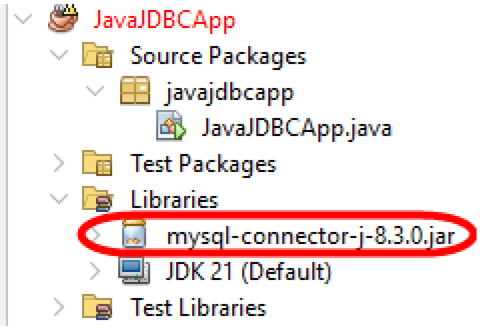
○ Relative Path: ../../../Downloads/mysql-connect

○ Path from Variable: <no suitable variabl

● Absolute Path: C:\Users\Nadeera\Downloads\my

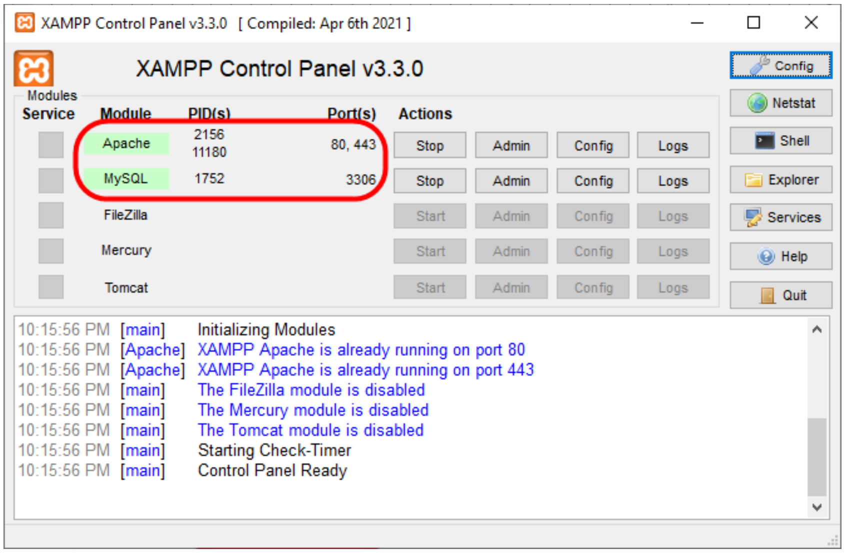
File Name: mysql-connector-j-8.3.0.jar

Files of Type: Classpath Entry (folder, ZIP or JAR file)



05. Create Database in a MariaDB Server

Start Apache and MariaDB Server

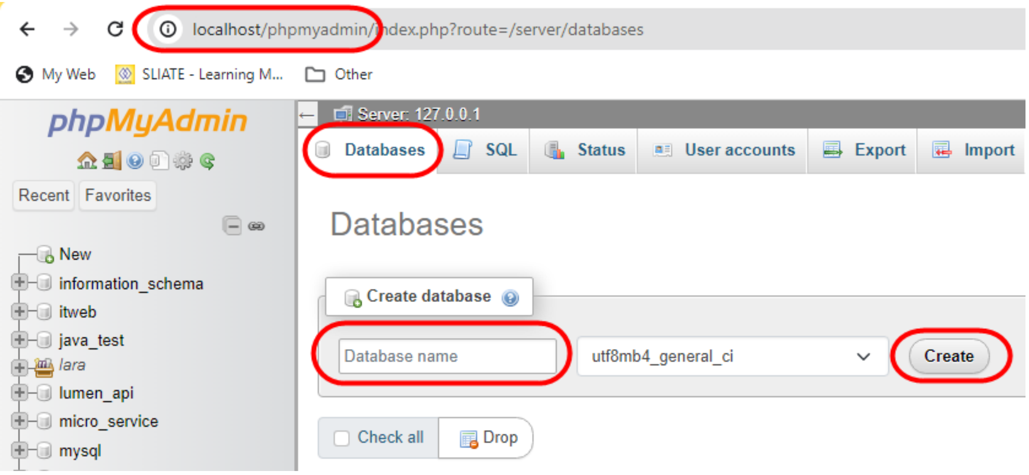

XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]
XAMPP Control Panel v3.3.0
Config
Netstat
Shell
Explorer
Services
Help
Quit

Modules Service	Module	PID(s)	Port(s)	Actions			
	Apache	2156 11180	80, 443	Stop	Admin	Config	Logs
	MySQL	1752	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
10:15:56 PM  [main]     Initializing Modules
10:15:56 PM  [Apache]   XAMPP Apache is already running on port 80
10:15:56 PM  [Apache]   XAMPP Apache is already running on port 443
10:15:56 PM  [main]     The FileZilla module is disabled
10:15:56 PM  [main]     The Mercury module is disabled
10:15:56 PM  [main]     The Tomcat module is disabled
10:15:56 PM  [main]     Starting Check-Timer
10:15:56 PM  [main]     Control Panel Ready
```

Open your Web Browser and enter the following address.

http://localhost/phpmyadmin and create a Database named “java_test”



Create a Table named “users”

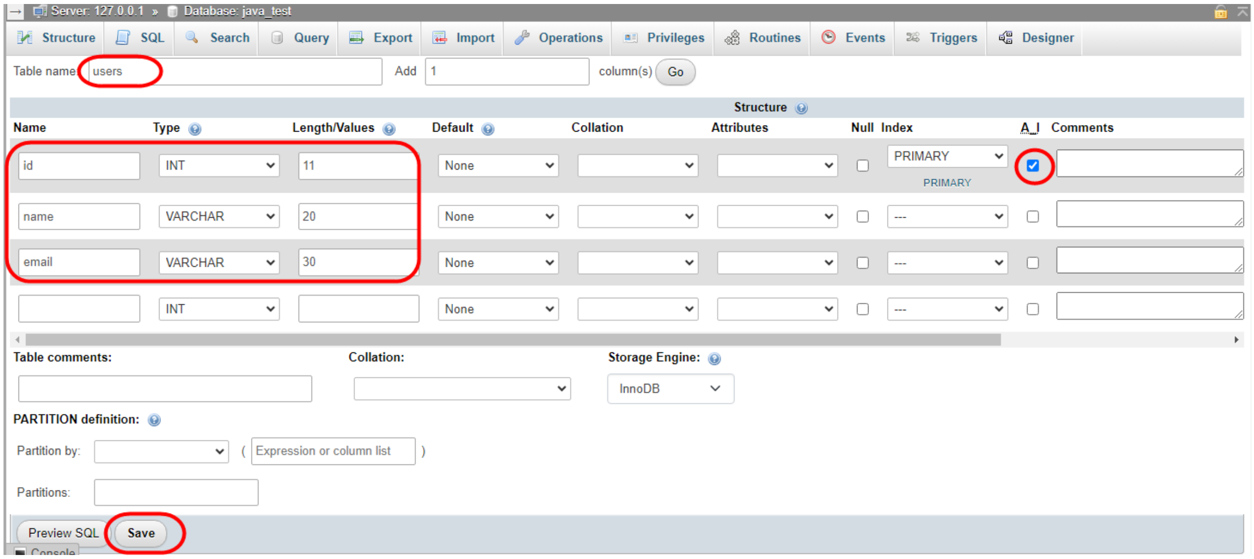


Table name: users Add 1 column(s) Go

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	Comments
id	INT	11	None			□	PRIMARY PRIMARY	☑	
name	VARCHAR	20	None			□	---	□	
email	VARCHAR	30	None			□	---	□	
	INT		None			□	---	□	

Table comments:

Collation:

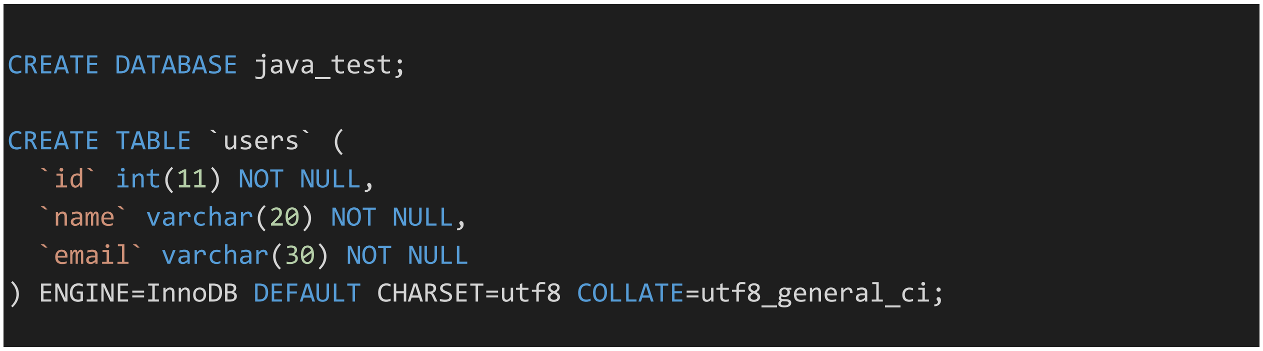
Storage Engine: InnoDB

PARTITION definition:

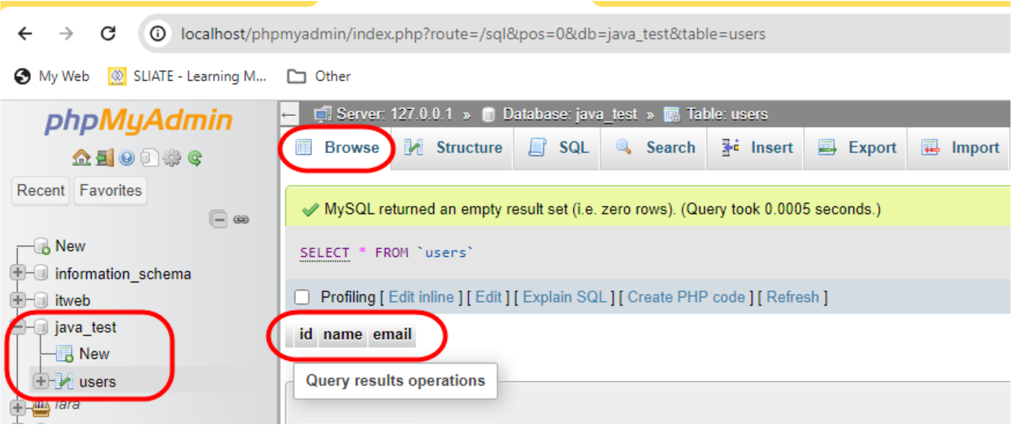
Partition by: (Expression or column list)

Partitions:

Following are the SQL statements for create a Database and the Table



```
CREATE DATABASE java_test;

CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `name` varchar(20) NOT NULL,
  `email` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;
```



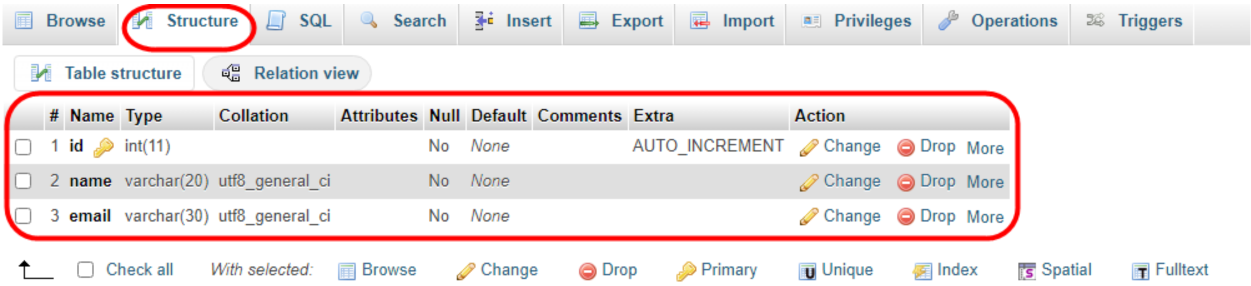
localhost/phpmyadmin/index.php?route=/sql&pos=0&db=java_test&table=users

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)

```
SELECT * FROM `users`
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

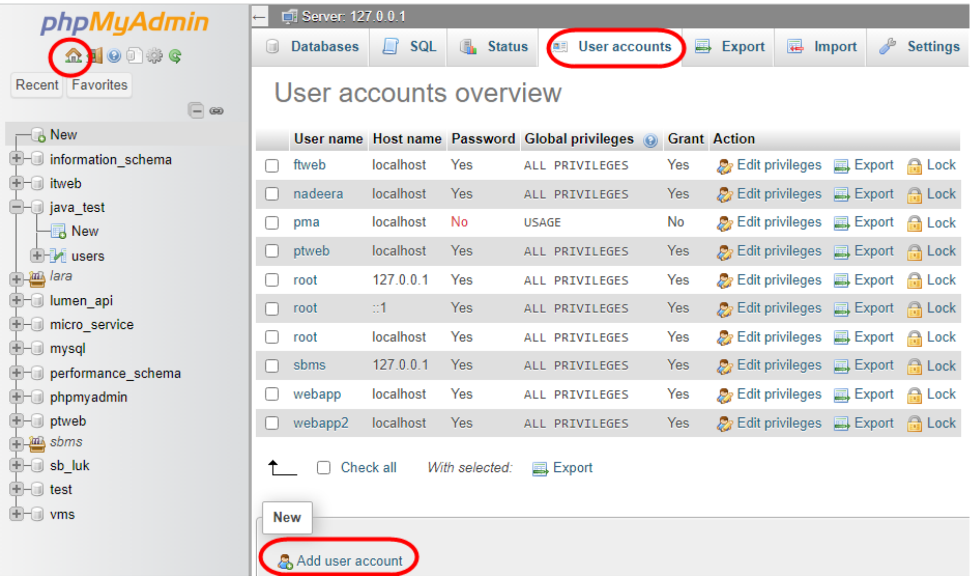
id	name	email



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	name	varchar(20)	utf8_general_ci		No	*None*			Change Drop More
3	email	varchar(30)	utf8_general_ci		No	*None*			Change Drop More

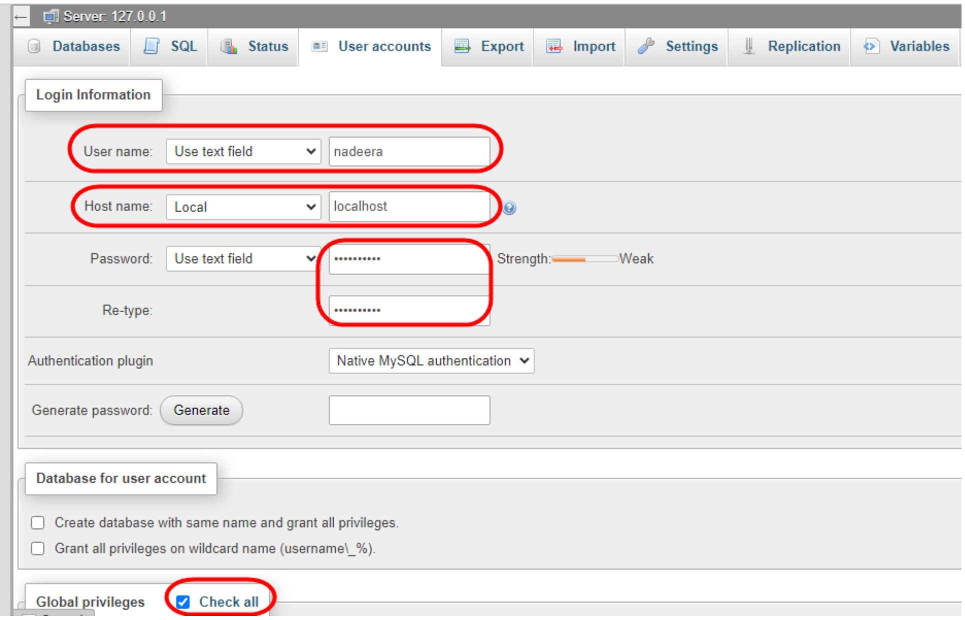
06. Create a Database User

Database users in DBMS can access the database and retrieve the data from the database using applications and interfaces provided by the Database Management System (DBMS).



User accounts overview

User name	Host name	Password	Global privileges	Grant	Action
ftweb	localhost	Yes	ALL PRIVILEGES	Yes	Edit privileges Export Lock
nadeera	localhost	Yes	ALL PRIVILEGES	Yes	Edit privileges Export Lock
pma	localhost	No	USAGE	No	Edit privileges Export Lock
ptweb	localhost	Yes	ALL PRIVILEGES	Yes	Edit privileges Export Lock
root	127.0.0.1	Yes	ALL PRIVILEGES	Yes	Edit privileges Export Lock
root	::1	Yes	ALL PRIVILEGES	Yes	Edit privileges Export Lock
root	localhost	Yes	ALL PRIVILEGES	Yes	Edit privileges Export Lock
sbms	127.0.0.1	Yes	ALL PRIVILEGES	Yes	Edit privileges Export Lock
webapp	localhost	Yes	ALL PRIVILEGES	Yes	Edit privileges Export Lock
webapp2	localhost	Yes	ALL PRIVILEGES	Yes	Edit privileges Export Lock



Login Information

User name:	Use text field	nadeera
Host name:	Local	localhost
Password:	Use text field	•••••••••• Strength: Weak
Re-type:		••••••••••
Authentication plugin		Native MySQL authentication
Generate password:	Generate	

Database for user account

□ Create database with same name and grant all privileges.

□ Grant all privileges on wildcard name (username_%).

Global privileges ☑ Check all

Add the global privileges to the user and click Go button or following SQL statements use to create user in a MariaDB Server

```
CREATE USER 'nadeera'@'localhost' IDENTIFIED VIA mysql_native_password
USING '***';GRANT ALL PRIVILEGES ON *.* TO 'nadeera'@'localhost'
REQUIRE NONE WITH GRANT OPTION MAX_QUERIES_PER_HOUR 0
MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;
```

nadeera = [your username]

*** = [your password]

07. Establish a Connection

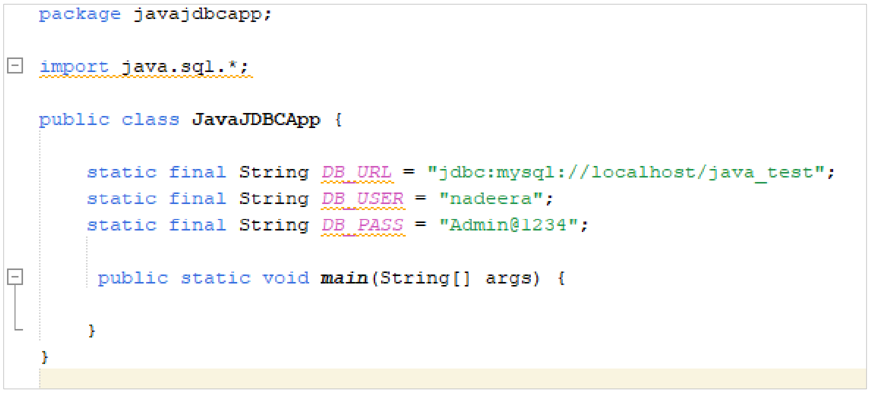
To use a class or a package from the library, you need to use the import keyword. By doing import java.sql.* you import all classes from the package java.sql at once, so that you don't have to import them one by one.

Import Java SQL Package

```
import java.sql.*;
```

Create three variables for connectivity.

```
static final String DB_URL = "jdbc:mysql://localhost/java_test";
static final String DB_USER = "nadeera";
static final String DB_PASS = "Admin@1234";

// java_test - is database
// nadeera - is database user
// Admin@1234- is password of the database user
```



```
package javajdbcapp;

import java.sql.*;

public class JavaJDBCApp {

    static final String DB_URL = "jdbc:mysql://localhost/java_test";
    static final String DB_USER = "nadeera";
    static final String DB_PASS = "Admin@1234";

     public static void main(String[] args) {

    }
}
```

08. Create a Method for Insert Data to the MySQL table

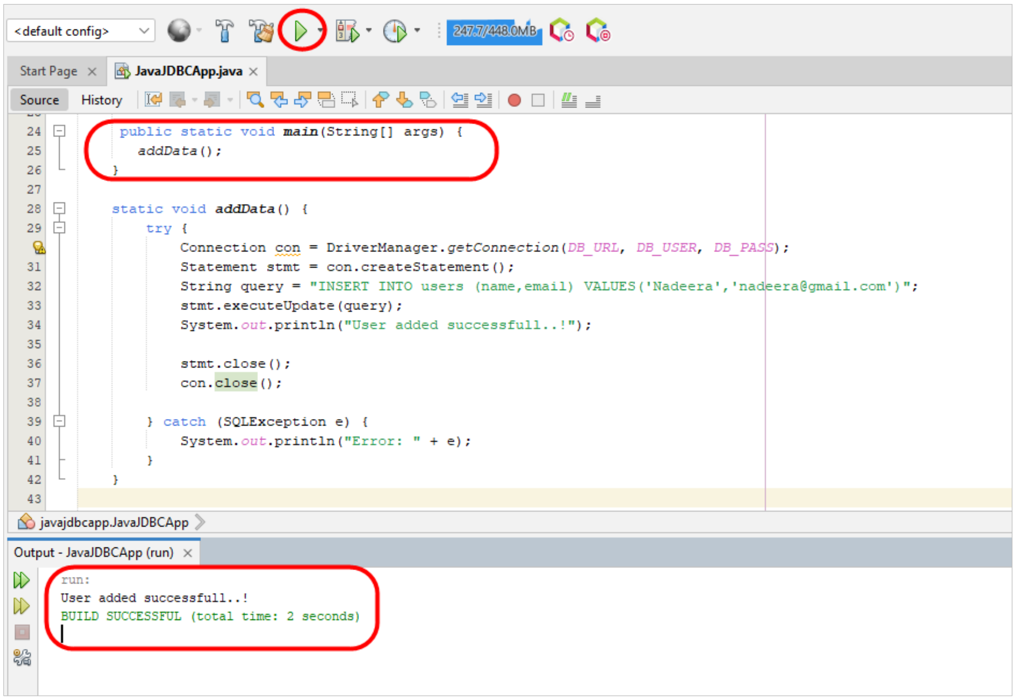
```
static void addData() {
      try {
         Connection con = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
         Statement stmt = con.createStatement();
         String query = "INSERT INTO users (name,email) VALUES('Nadeera','nadeera@gmail.com')";
         stmt.executeUpdate(query);
         System.out.println("User added successfull..!");

         con.close();

      } catch (SQLException e) {
         System.out.println("Error: " + e);
      }
   }
```

A Connection is a session between a Java application and a database. It helps to establish a connection with the database.

Once a connection is established you can interact with the database. The Statement interface provides methods to execute queries with the database. The important methods of Statement interface are as follows:

- **executeQuery(String sql)** is used to execute SELECT query. It returns the object of ResultSet.
- **executeUpdate(String sql)** is used to execute specified query, it may be create, drop, insert, update, delete etc.

After executing SQL statement you need to close the connection and release the session. The close() method of Connection interface is used to close the connection.



```
    public static void main(String[] args) {
        addData();
    }

    static void addData() {
        try {
            Connection con = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
            Statement stmt = con.createStatement();
            String query = "INSERT INTO users (name,email) VALUES('Nadeera','nadeera@gmail.com')";
            stmt.executeUpdate(query);
            System.out.println("User added successfull..!");

            stmt.close();
            con.close();

        } catch (SQLException e) {
            System.out.println("Error: " + e);
        }
    }
```

```
run:
User added successfull..!
BUILD SUCCESSFUL (total time: 2 seconds)
```

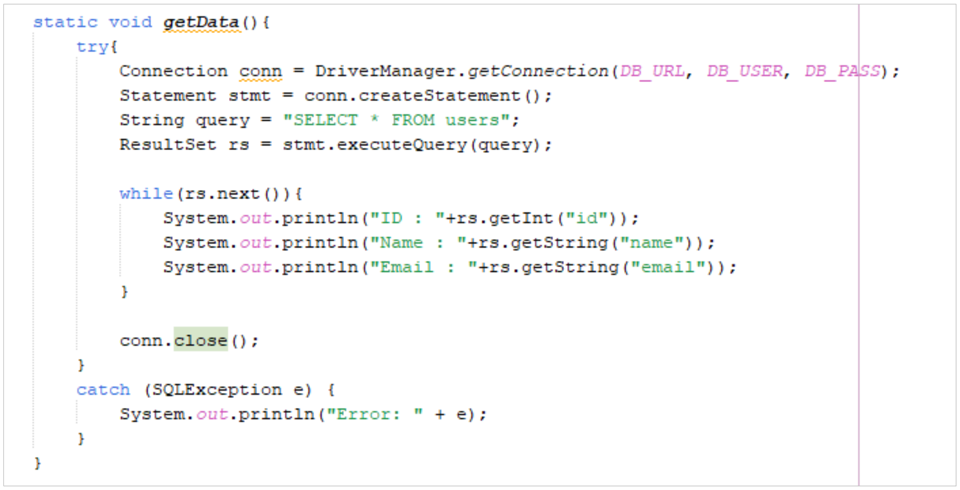
09. Create a Method for Select Data to from MySQL table

```
static void getData(){
    try{
      Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
      Statement stmt = conn.createStatement();
      String query = "SELECT * FROM users";
      ResultSet rs = stmt.executeQuery(query);

      while(rs.next()){
        System.out.println("ID : "+rs.getInt("id"));
        System.out.println("Name : "+rs.getString("name"));
        System.out.println("Email : "+rs.getString("email"));
      }

      conn.close();
    }
    catch (SQLException e) {
      System.out.println("Error: " + e);
    }
  }
```

The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row.

Commonly used methods of ResultSet interface

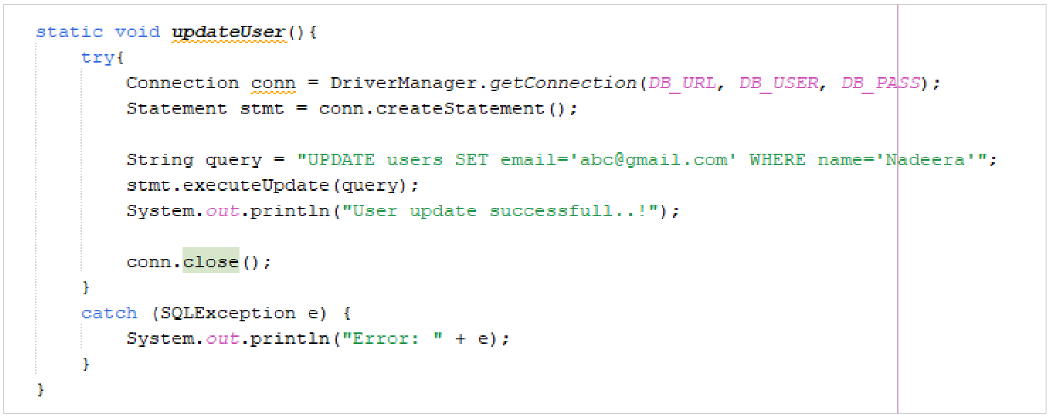
public boolean next():	is used to move the cursor to the one row next from the current position.
public boolean previous():	is used to move the cursor to the one row previous from the current position.
public boolean first():	is used to move the cursor to the first row in result set object.
public boolean last():	is used to move the cursor to the last row in result set object.
public int getInt(int columnIndex):	is used to return the data of specified column index of the current row as int.
public int getInt(String columnName):	is used to return the data of specified column name of the current row as int.
public String getString(int columnIndex):	is used to return the data of specified column index of the current row as String.
public String getString(String columnName):	is used to return the data of specified column name of the current row as String.



```
static void getData(){
    try{
        Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
        Statement stmt = conn.createStatement();
        String query = "SELECT * FROM users";
        ResultSet rs = stmt.executeQuery(query);

        while(rs.next()){
            System.out.println("ID : "+rs.getInt("id"));
            System.out.println("Name : "+rs.getString("name"));
            System.out.println("Email : "+rs.getString("email"));
        }

        conn.close();
    }
    catch (SQLException e) {
        System.out.println("Error: " + e);
    }
}
```

10. Create a Method for Update Data

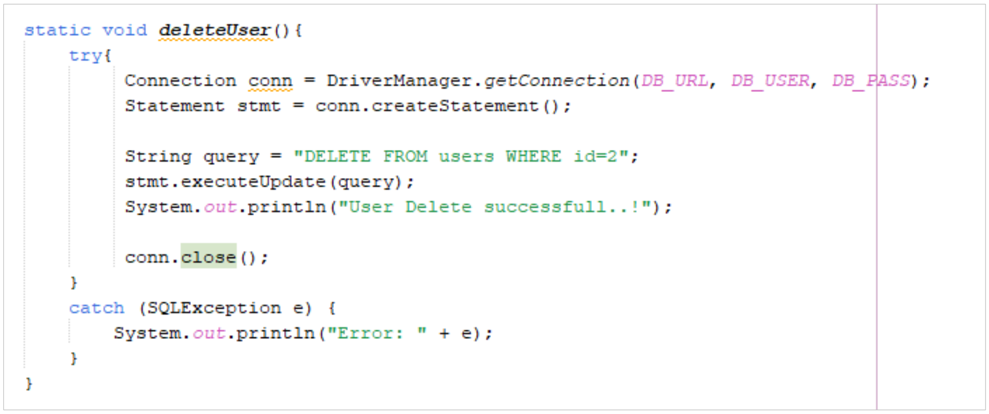
```
static void updateUser(){
    try{
      Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
      Statement stmt = conn.createStatement();

      String query = "UPDATE users SET email='abc@gmail.com' WHERE name='Nadeera'";
      stmt.executeUpdate(query);
      System.out.println("User update successfull..!");

      stmt.close();
      conn.close();
    }
    catch (SQLException e) {
      System.out.println("Error: " + e);
    }
  }
```



```
static void updateUser(){
    try{
        Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
        Statement stmt = conn.createStatement();

        String query = "UPDATE users SET email='abc@gmail.com' WHERE name='Nadeera'";
        stmt.executeUpdate(query);
        System.out.println("User update successfull..!");

        conn.close();
    }
    catch (SQLException e) {
        System.out.println("Error: " + e);
    }
}
```

11. Create a Method for Delete Data

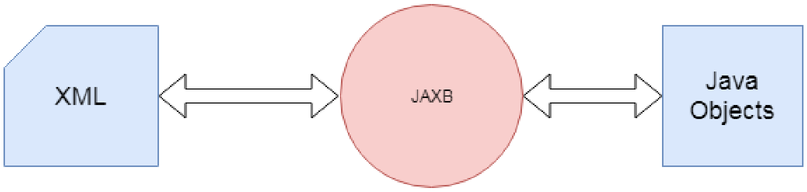
```
static void deleteUser(){
    try{
        Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
        Statement stmt = conn.createStatement();

        String query = "DELETE FROM users WHERE id=2";
        stmt.executeUpdate(query);
        System.out.println("User Delete successfull..!");

        stmt.close();
        conn.close();
    }
    catch (SQLException e) {
      System.out.println("Error: " + e);
    }
  }
```



```
static void deleteUser(){
    try{
        Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
        Statement stmt = conn.createStatement();

        String query = "DELETE FROM users WHERE id=2";
        stmt.executeUpdate(query);
        System.out.println("User Delete successfull..!");

        conn.close();
    }
    catch (SQLException e) {
        System.out.println("Error: " + e);
    }
}
```

JAXB

JAXB stands for Java Architecture for XML Binding. It provides mechanism to marshal (write) java objects into XML and unmarshal (read) XML into object. Simply, you can say it is used to convert java object into xml and vice-versa.



It supports a binding framework that maps XML elements and attributes to Java fields and properties using Java annotations.

JAXB uses Java annotations to augment the generated classes with additional information. Adding such annotations to existing Java classes prepares them for the JAXB runtime.

JAXB Annotations

JAXB uses Java annotations to augment the generated classes with additional information. Adding such annotations to existing Java classes prepares them for the JAXB runtime.

* Java Annotation is a tag that represents the metadata i.e. attached with class, interface, methods or fields to indicate some additional

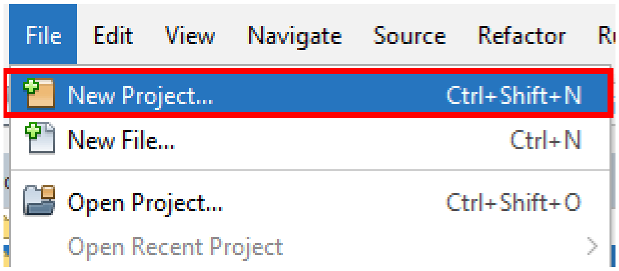
@XmlRootElement	: The name of the root XML element is derived from the class name, and we can also specify the name of the root element of the XML using its name attribute.
@XmlType	: define the order in which the fields are written in the XML file
@XmlElement	: define the actual XML element name that will be used
@XmlAttribute	: define the id field is mapped as an attribute instead of an element
@XmlTransient	: annotate fields that we don’t want to be included in XML

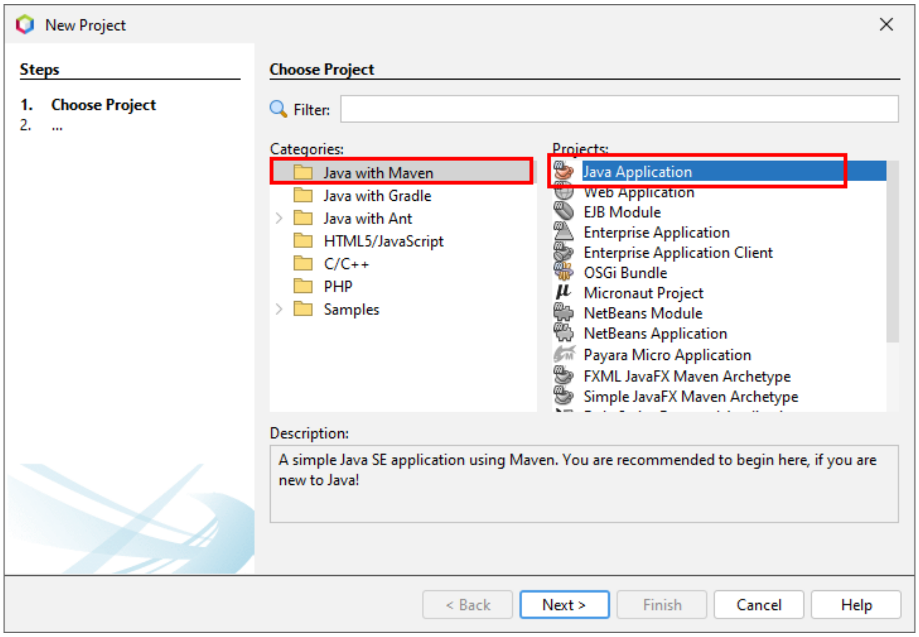
For more details on JAXB annotation, check out this link.

http://docs.oracle.com/javaee/7/api/javax/xml/bind/annotation/package-summary.html

Read and Write XML File Using JAXB

1. **Create Java Projects with Maven using NetBeans IDE**

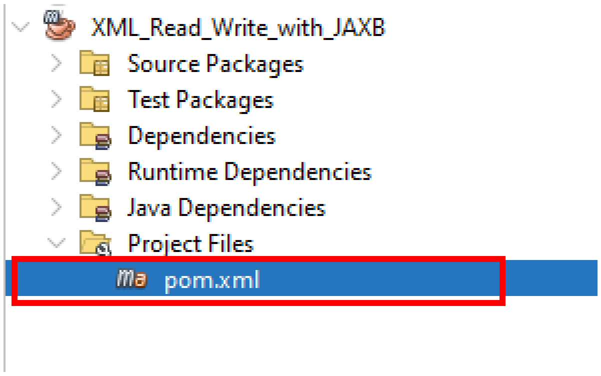




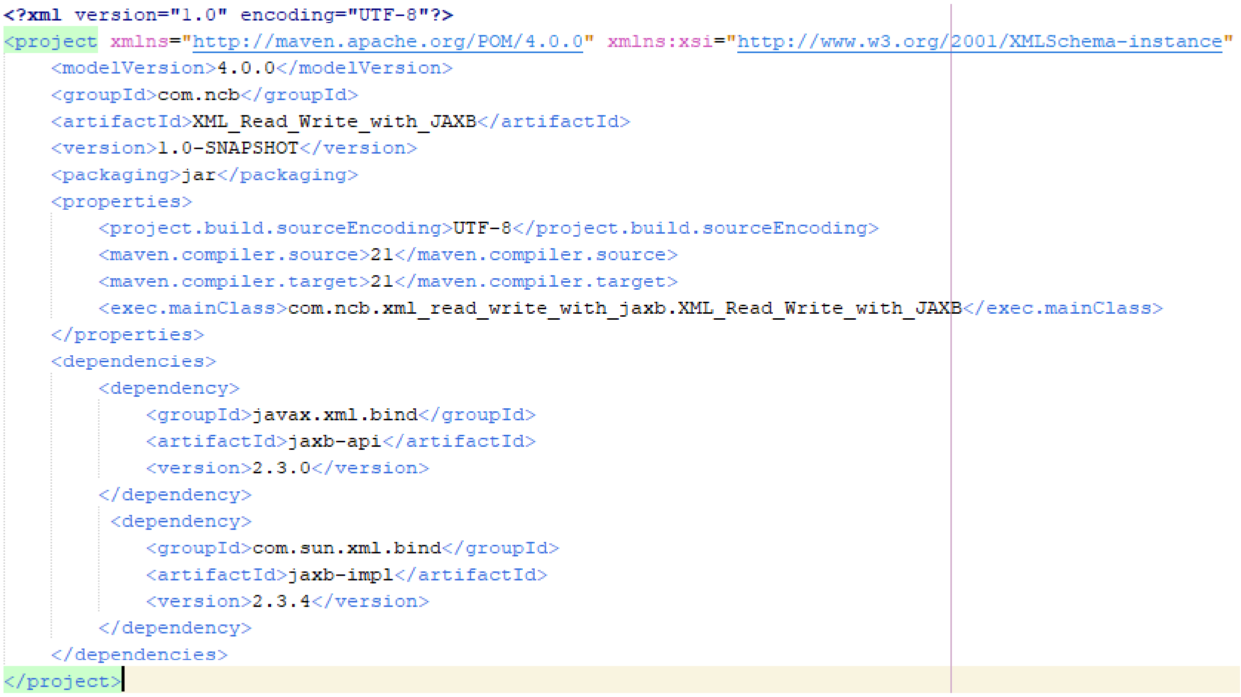
Description:

A simple Java SE application using Maven. You are recommended to begin here, if you are new to Java!

2. **Add JAXB and its Dependencies to POM.xml**



```
<dependencies>
        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
            <version>2.3.0</version>
        </dependency>
         <dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>2.3.4</version>
        </dependency>
</dependencies>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.ncb</groupId>
    <artifactId>XML_Read_Write_with_JAXB</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <exec.mainClass>com.ncb.xml_read_write_with_jaxb.XML_Read_Write_with_JAXB</exec.mainClass>
    </properties>
    <dependencies>
        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
            <version>2.3.0</version>
        </dependency>
         <dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>2.3.4</version>
        </dependency>
    </dependencies>
</project>
```

3. Create a new class named "Student."

```
import javax.xml.bind.annotation.*;

@XmlType(propOrder = { "name", "batch" })

    public class Student {
    private String regno;
    private String name;
    private int batch;
    private String ati;

    public String getRegno() {
        return regno;
    }

    @XmlAttribute
    public void setRegno(String regno) {
        this.regno = regno;
    }

    public String getName() {
        return name;
    }

    @XmlElement(name = "name")
    public void setName(String name) {
        this.name = name;
    }

    public int getBatch() {
        return batch;
    }

    @XmlElement(name = "batch")
    public void setBatch(int batch) {
        this.batch = batch;
    }

    public String getAti() {
        return ati;
    }

    @XmlTransient
    public void setAti(String ati) {
        this.ati = ati;
    }
}
```

4. **Create another new class named " SLIATE."**

```
import java.util.ArrayList;
import javax.xml.bind.annotation.*;

@XmlRootElement(name = "sliate")
@XmlAccessorType(XmlAccessType.FIELD)

public class SLIATE {

  @XmlElement(name = "student")
  private ArrayList<Student> studentList;

    public ArrayList<Student> getStudentList() {
        return studentList;
    }

    public void setStudentList(ArrayList<Student> studentList) {
        this.studentList = studentList;
    }
}
```

5. **Import following packages to your main Class**

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
```

6. Create a Method for Write a XML File

```
public static void marshal(){
    try{

      SLIATE kandyATI = new SLIATE();
      ArrayList<Student> lst=new ArrayList<>();

      Student stu1 = new Student();
      stu1.setRegno("PT001");
      stu1.setName("Nadeera");
      stu1.setBatch(2023);
      stu1.setAti("Kandy");
      lst.add(stu1);

      Student stu2 = new Student();
      stu2.setRegno("PT002");
      stu2.setName("Nuwan");
      stu2.setBatch(2023);
      stu2.setAti("Kandy");
      lst.add(stu2);

      kandyATI.setStudentList(lst);

      JAXBContext context = JAXBContext.newInstance(SLIATE.class);
      Marshaller mar= context.createMarshaller();
      mar.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
      mar.marshal(kandyATI, new File("./ati_students.xml"));
    }
    catch(JAXBException ex){
      System.out.println("Error!:"+ex);
    }
  }
```

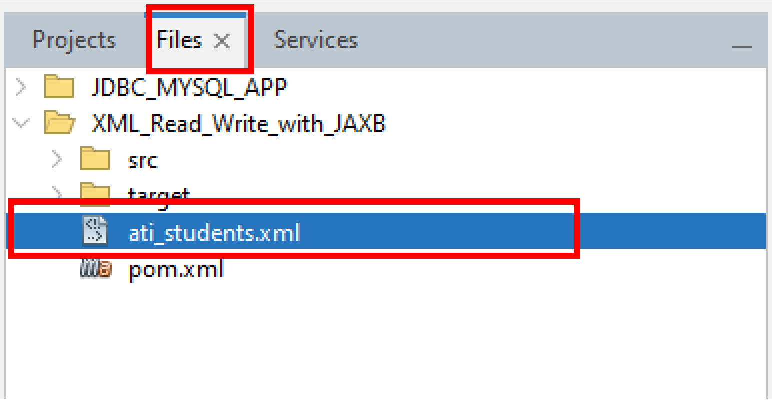
7. Create a Method for Read the XML File

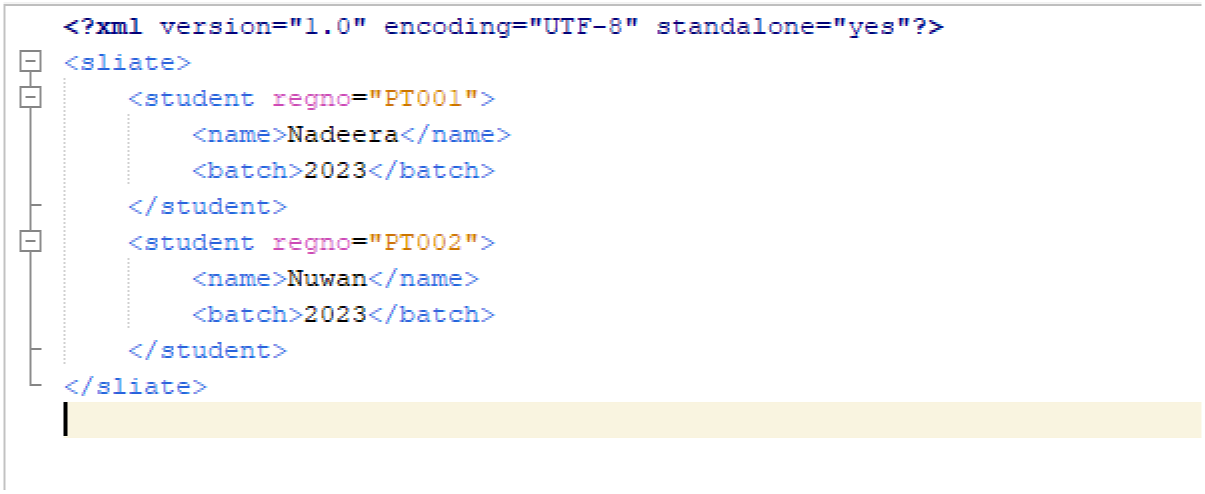
```
public static void unmarshal() {
        try {
            JAXBContext context = JAXBContext.newInstance(SLIATE.class);
            Unmarshaller um = context.createUnmarshaller();

            SLIATE sliate = (SLIATE) um.unmarshal(new FileReader("./ati_students.xml"));
            ArrayList<Student> list = sliate.getStudentList();

            for (Student stu : list) {
                System.out.println("Name: " + stu.getName() + " Batch "
                        + stu.getBatch());
            }

        } catch (JAXBException | IOException ex) {
            System.out.println("Error!:" + ex);
        }
    }
```

8. Invoke Write and Read Methods

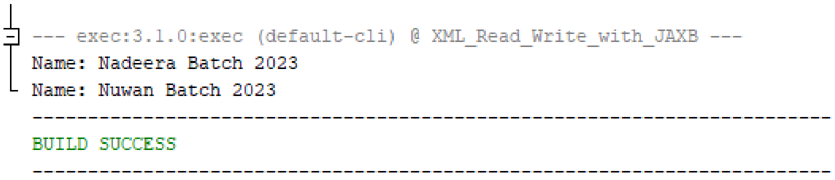
```
public static void main(String[] args) {
        marshal();
        unmarshal();
    }
```

9. Navigate to the Files Tab to view the created XML file.





```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sliate>
    <student regno="PT001">
        <name>Nadeera</name>
        <batch>2023</batch>
    </student>
    <student regno="PT002">
        <name>Nuwan</name>
        <batch>2023</batch>
    </student>
</sliate>
```



```
--- exec:3.1.0:exec (default-cli) @ XML_Read_Write_with_JAXB ---
Name: Nadeera Batch 2023
Name: Nuwan Batch 2023
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

Further Reading:

https://www.tutorialspoint.com/java_xml/index.htm

https://www.javatpoint.com/jaxb-tutorial

HNDIT

XML

Enterprise Architecture

DWNC. Bandaranayake

What is XML

XML stands for Extensible Markup Language. It is a text-based markup language derived from Standard Generalized Markup Language (SGML).

- XML was designed to store and transport data.
- XML was designed to be both human- and machine-readable.
- A language for describing data
- A tool for building new data description languages
- Open to define in any way you want
- A markup language. That means that “marks” or tags are used to identify the data elements. Data and tags are generally stored as plain text.

What is XML Used For?

XML is one of the most widely-used formats for sharing structured information today:

- Between programs
- Between people
- Between computers and people
- Both locally and across networks

XML Syntax

XML is strict on formatting; if the formatting is off, programs that process or display the encoded data will return an error.

The XML Prolog

The XML document can optionally have an XML declaration. It is called as the XML prolog.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- The XML declaration is case sensitive and must begin with "<?xml>" where "xml" is written in lower-case.

- If document contains XML declaration, then it strictly needs to be the first statement of the XML document.

- The XML declaration strictly needs be the first statement in the XML document.

- An HTTP protocol can override the value of encoding that you put in the XML declaration.

Tags and Elements

An XML file is structured by several XML-elements, also called XML-nodes or XML-tags. The names of XML-elements are enclosed in triangular brackets < > as shown below

```
<element>....</element>
```

Root Element

An XML document can have only one root element.

```
<root>
  <x>...</x>
  <y>...</y>
</root>
```

All XML Elements Must Have a Closing Tag

In XML, it is illegal to omit the closing tag. All elements must have a closing tag except prolog. XML prolog does not have a closing tag! This is not an error. The prolog is not a part of the XML document.

```
<message>This is correct</message>
```

XML Tags are Case Sensitive

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.

XML Elements Must be Properly Nested

"Properly nested" simply means that since the <i> element is opened inside the <b> element, it must be closed inside the <b> element.

```
<b><i>This text is bold and italic</i></b>
```

XML Attribute Values Must Always be Quoted

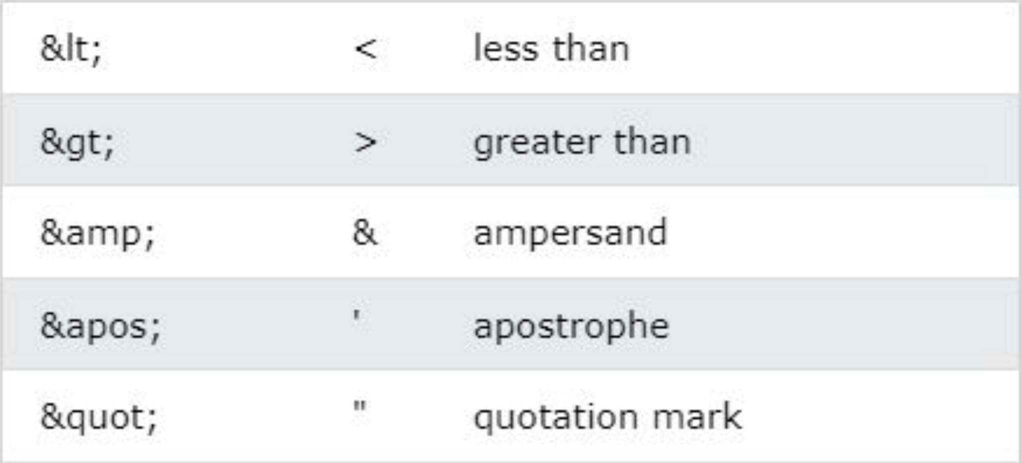
XML elements can have attributes in name/value pairs just like in HTML. In XML, the attribute values must always be quoted:

```
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

Entity References

Some characters have a special meaning in XML. If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

To avoid this error, replace the "<" character with an entity reference:



&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Comments in XML

The syntax for writing comments in XML is similar to that of HTML.

```
<!-- This is a comment -->
```

XML Naming Rules

XML elements must follow these naming rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

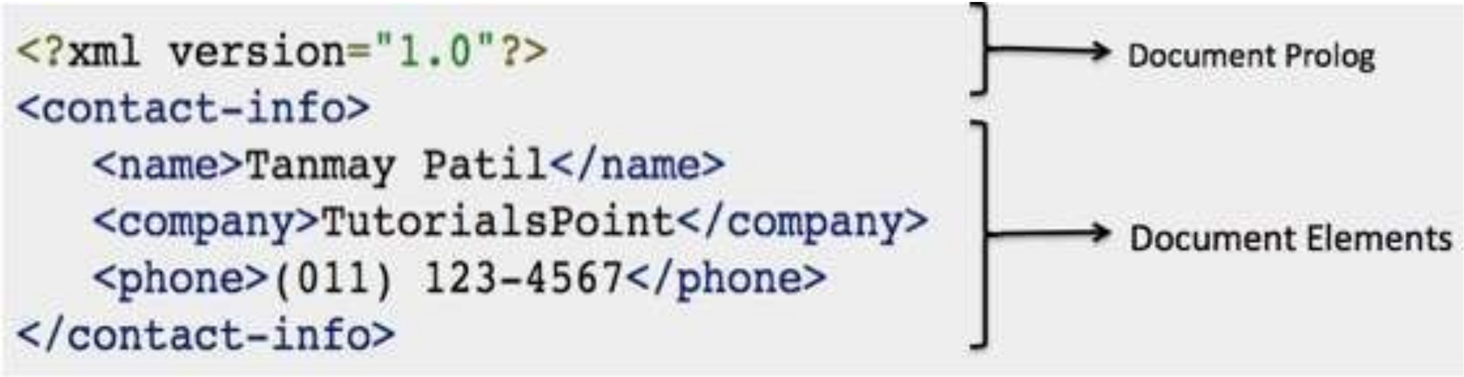
Create descriptive, short and simple names.

```
<book_title> not like this: <the_title_of_the_book>
```

- Avoid "-"
- Avoid "."
- Avoid ":"
- Non-English letters

XML Document Example

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```



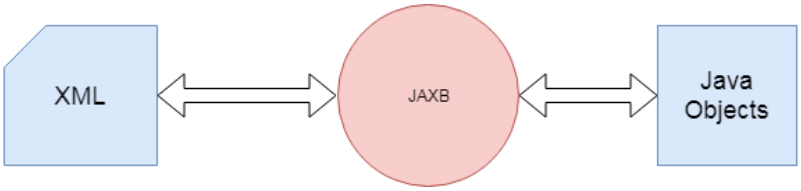
```
<?xml version="1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
<book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

END

JAXB

JAXB stands for Java Architecture for XML Binding. It provides mechanism to marshal (write) java objects into XML and unmarshal (read) XML into object. Simply, you can say it is used to convert java object into xml and vice-versa.



It supports a binding framework that maps XML elements and attributes to Java fields and properties using Java annotations.

JAXB uses Java annotations to augment the generated classes with additional information. Adding such annotations to existing Java classes prepares them for the JAXB runtime.

JAXB Annotations

JAXB uses Java annotations to augment the generated classes with additional information. Adding such annotations to existing Java classes prepares them for the JAXB runtime.

* Java Annotation is a tag that represents the metadata i.e. attached with class, interface, methods or fields to indicate some additional

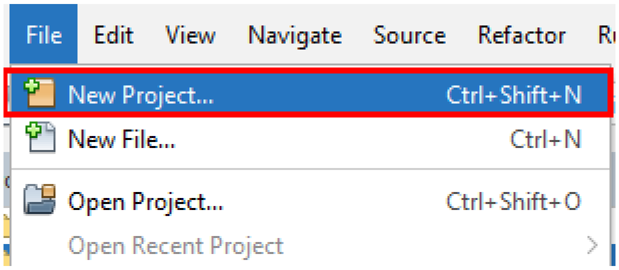
@XmlRootElement	: The name of the root XML element is derived from the class name, and we can also specify the name of the root element of the XML using its name attribute.
@XmlType	: define the order in which the fields are written in the XML file
@XmlElement	: define the actual XML element name that will be used
@XmlAttribute	: define the id field is mapped as an attribute instead of an element
@XmlTransient	: annotate fields that we don’t want to be included in XML

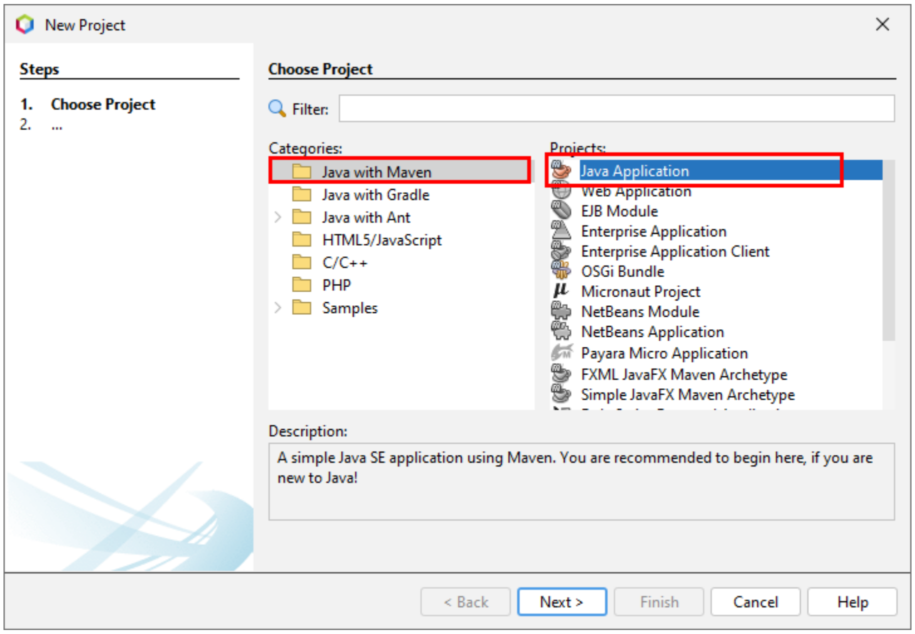
For more details on JAXB annotation, check out this link.

http://docs.oracle.com/javaee/7/api/javax/xml/bind/annotation/package-summary.html

Read and Write XML File Using JAXB

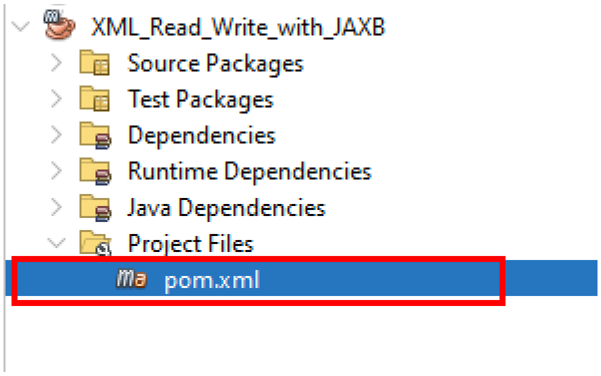
01. Create Java Projects with Maven using NetBeans IDE



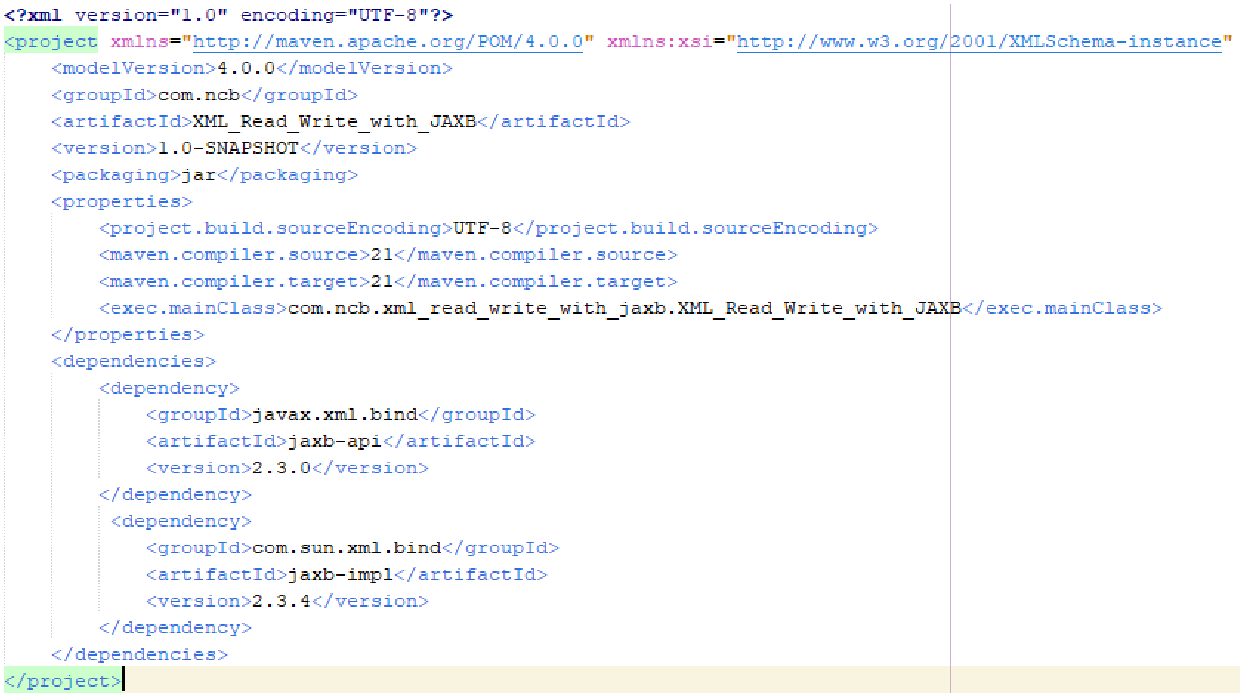


A simple Java SE application using Maven. You are recommended to begin here, if you are new to Java!

02. Add JAXB and its Dependencies to POM.xml



```
<dependencies>
        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
            <version>2.3.0</version>
        </dependency>
         <dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>2.3.4</version>
        </dependency>
</dependencies>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.ncb</groupId>
    <artifactId>XML_Read_Write_with_JAXB</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <exec.mainClass>com.ncb.xml_read_write_with_jaxb.XML_Read_Write_with_JAXB</exec.mainClass>
    </properties>
    <dependencies>
        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
            <version>2.3.0</version>
        </dependency>
         <dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>2.3.4</version>
        </dependency>
    </dependencies>
</project>
```

03. Create a new class named "Student."

```
import javax.xml.bind.annotation.*;

@XmlType(propOrder = { "name", "batch" })

   public class Student {
   private String regno;
   private String name;
   private int batch;
   private String ati;

   public String getRegno() {
      return regno;
   }

   @XmlAttribute
   public void setRegno(String regno) {
      this.regno = regno;
   }

   public String getName() {
      return name;
   }

   @XmlElement(name = "name")
   public void setName(String name) {
      this.name = name;
   }

   public int getBatch() {
      return batch;
   }

   @XmlElement(name = "batch")
   public void setBatch(int batch) {
      this.batch = batch;
   }

   public String getAti() {
      return ati;
   }

   @XmlTransient
   public void setAti(String ati) {
      this.ati = ati;
   }
}
```

04. Create another new class named " SLIATE."

```
import java.util.ArrayList;
import javax.xml.bind.annotation.*;

@XmlRootElement(name = "sliate")
@XmlAccessorType(XmlAccessType.FIELD)

public class SLIATE {

  @XmlElement(name = "student")
  private ArrayList<Student> studentList;

    public ArrayList<Student> getStudentList() {
        return studentList;
    }

    public void setStudentList(ArrayList<Student> studentList) {
        this.studentList = studentList;
    }
}
```

05. Import following packages to your main Class

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
```

06. Create a Method for Write a XML File

```
public static void marshal(){
    try{

      SLIATE kandyATI = new SLIATE();
      ArrayList<Student> lst=new ArrayList<>();

      Student stu1 = new Student();
      stu1.setRegno("PT001");
      stu1.setName("Nadeera");
      stu1.setBatch(2023);
      stu1.setAti("Kandy");
      lst.add(stu1);

      Student stu2 = new Student();
      stu2.setRegno("PT002");
      stu2.setName("Nuwan");
      stu2.setBatch(2023);
      stu2.setAti("Kandy");
      lst.add(stu2);

      kandyATI.setStudentList(lst);

      JAXBContext context = JAXBContext.newInstance(SLIATE.class);
      Marshaller mar= context.createMarshaller();
      mar.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
      mar.marshal(kandyATI, new File("./ati_students.xml"));
    }
    catch(JAXBException ex){
      System.out.println("Error!:"+ex);
    }
  }
```

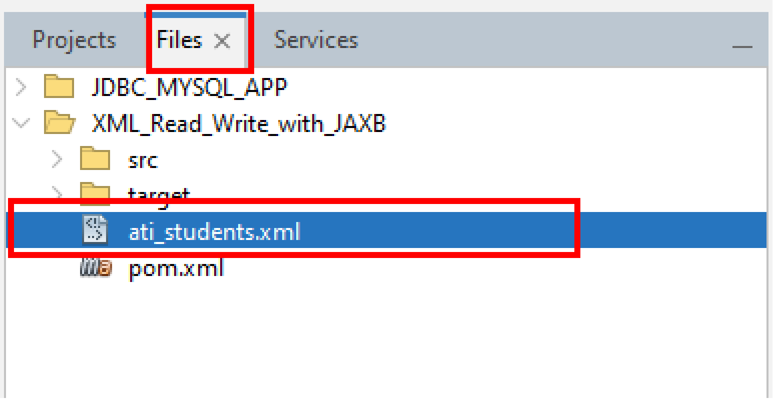
07. Create a Method for Read the XML File

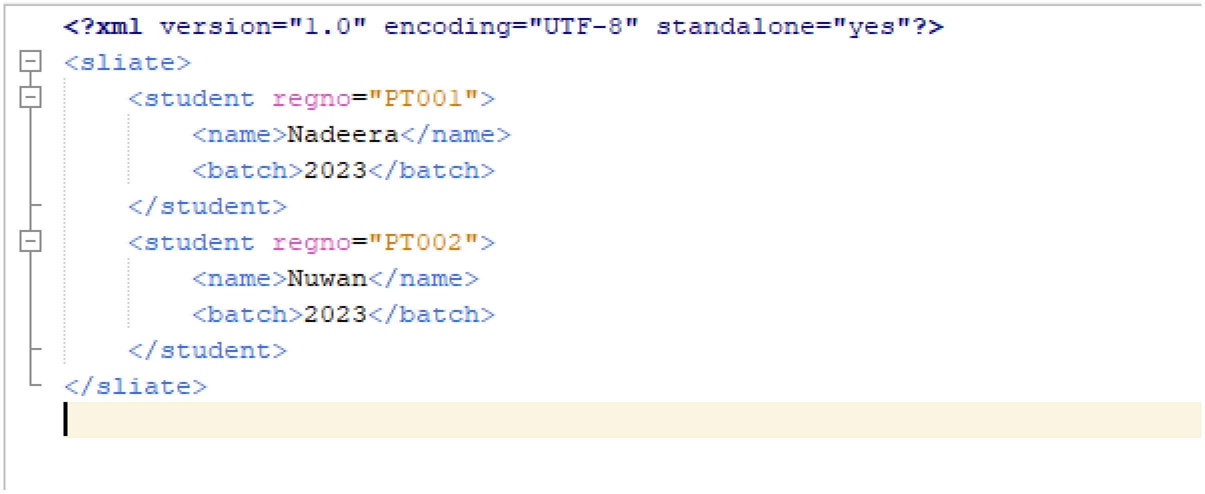
```
public static void unmarshal() {
      try {
        JAXBContext context = JAXBContext.newInstance(SLIATE.class);
        Unmarshaller um = context.createUnmarshaller();

        SLIATE sliate = (SLIATE) um.unmarshal(new FileReader("./ati_students.xml"));
        ArrayList<Student> list = sliate.getStudentList();

        for (Student stu : list) {
          System.out.println("Name: " + stu.getName() + " Batch "
                  + stu.getBatch());
        }

      } catch (JAXBException | IOException ex) {
        System.out.println("Error!:" + ex);
      }
  }
```

08. Invoke Write and Read Methods

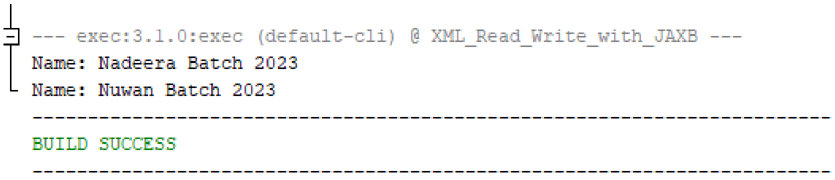
```
public static void main(String[] args) {
       marshal();
       unmarshal();
  }
```

09. Navigate to the Files Tab to view the created XML file.





```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sliate>
    <student regno="PT001">
        <name>Nadeera</name>
        <batch>2023</batch>
    </student>
    <student regno="PT002">
        <name>Nuwan</name>
        <batch>2023</batch>
    </student>
</sliate>
```



```
--- exec:3.1.0:exec (default-cli) @ XML_Read_Write_with_JAXB ---
Name: Nadeera Batch 2023
Name: Nuwan Batch 2023
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

Further Reading:

https://www.tutorialspoint.com/java_xml/index.htm

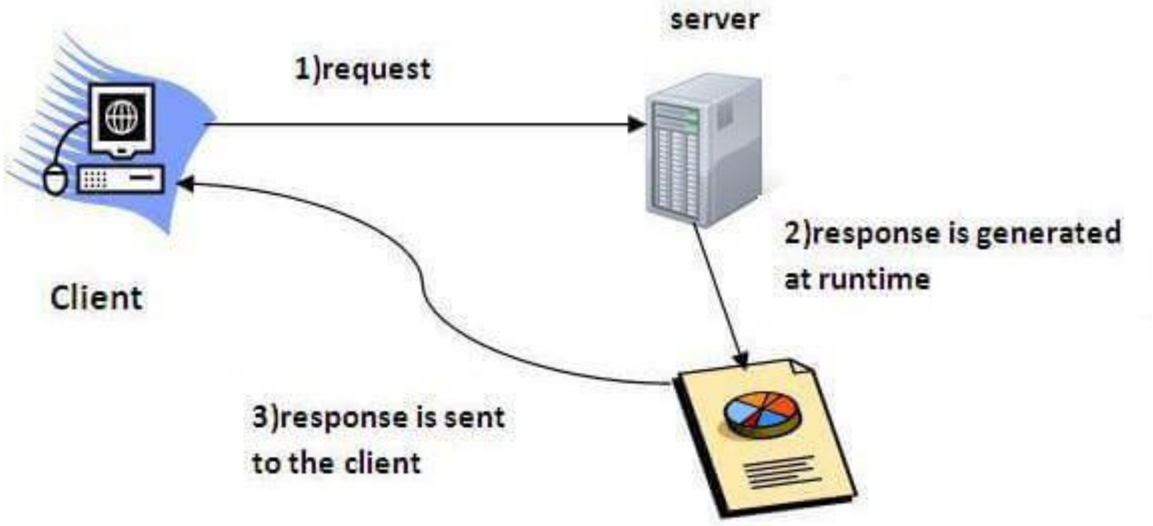
https://www.javatpoint.com/jaxb-tutorial

SERVLETS

What is a Servlet?

Servlet can be described in many ways, depending on the context.

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.

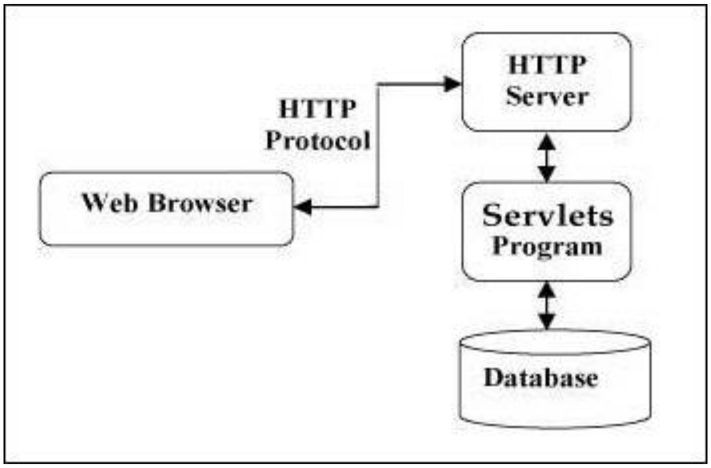

server
1)request
Client
2)response is generated at runtime
3)response is sent to the client

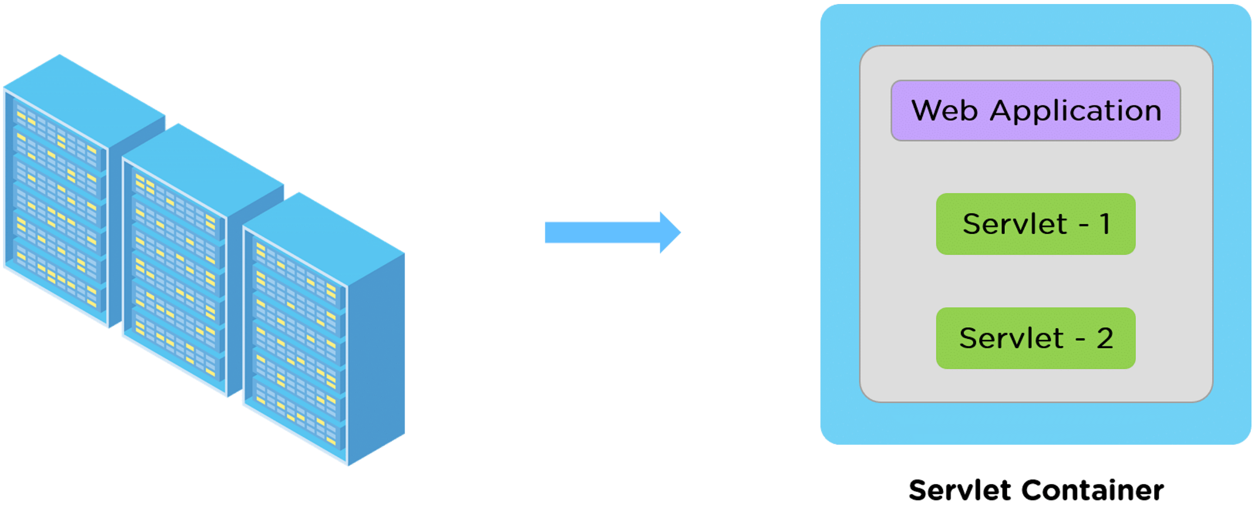
What is a web application?

A web application is an application accessible from the web. A web application is composed of web components like Servlet, JSP, Filter, etc. and other elements such as HTML, CSS, and JavaScript. The web components typically execute in Web Server and respond to the HTTP request.

Servlets Architecture

The following diagram shows the position of Servlets in a Web Application.





The Java Servlet Life cycle includes three stages right from its start to the end until the Garbage Collector clears it. These three stages are described below.

1. The servlet is initialized by calling the init() method.
2. The servlet calls service() method to process a client's request.
3. The servlet is terminated by calling the destroy() method.

The init() Method

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations

The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server generate a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

The service () method is called by the container and service method invokes doGet, doPost, doPut, doDelete, etc. methods as appropriate. So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.

The doGet() and doPost() are most frequently used methods with in each service request. Here is the signature of these two methods.

The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. After the destroy() method is called, the servlet object is marked for garbage collection.

The destory() will be called :

1.when the container shuts down or the application shuts down;

2.when the container decides that there is a shortage of memory;

3.when this servlet hasn't got a request in a long time.

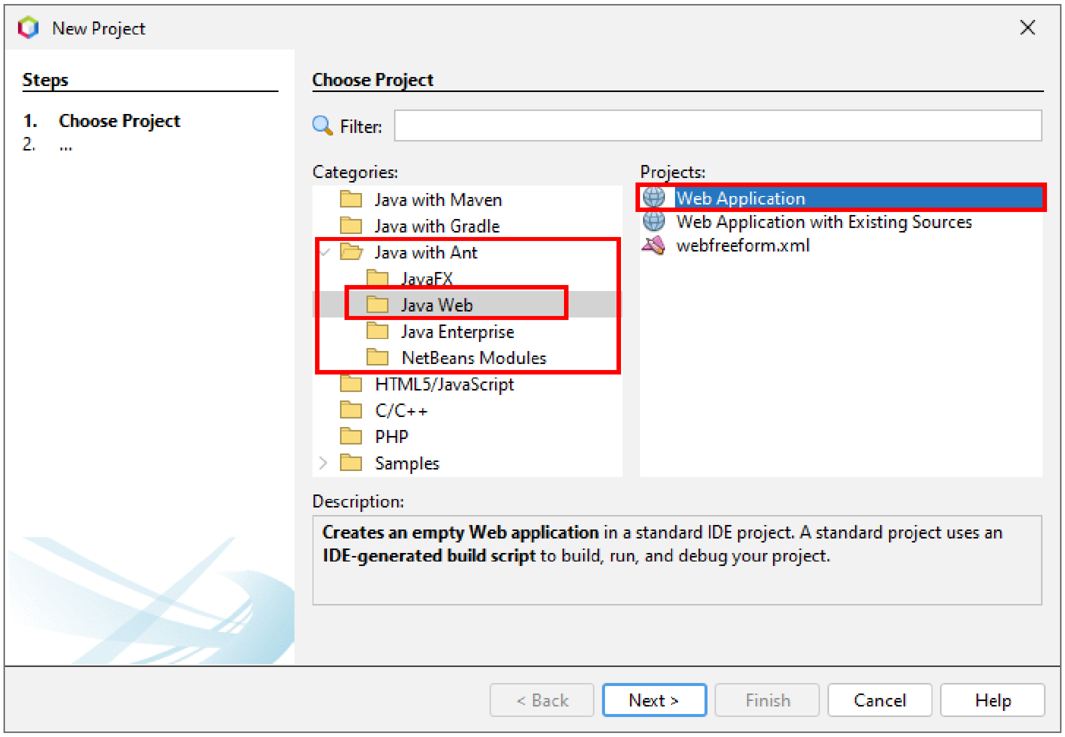
Servlets are Java classes which service HTTP requests and implement the javax.servlet.Servlet interface. Web application developers typically write servlets that extend javax.servlet.http.HttpServlet, an abstract class that implements the Servlet interface and is specially designed to handle HTTP requests.

Following is the sample source code structure of a servlet example to show Hello World

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloWorld extends HttpServlet {

  private String message;

  public void init() throws ServletException {
    // Do required initialization
    message = "Hello World";
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    // Set response content type
    response.setContentType("text/html");

    // Actual logic goes here.
    PrintWriter out = response.getWriter();
    out.println("<h1>" + message + "</h1>");
  }

  public void destroy() {
    // do nothing.
  }
}
```

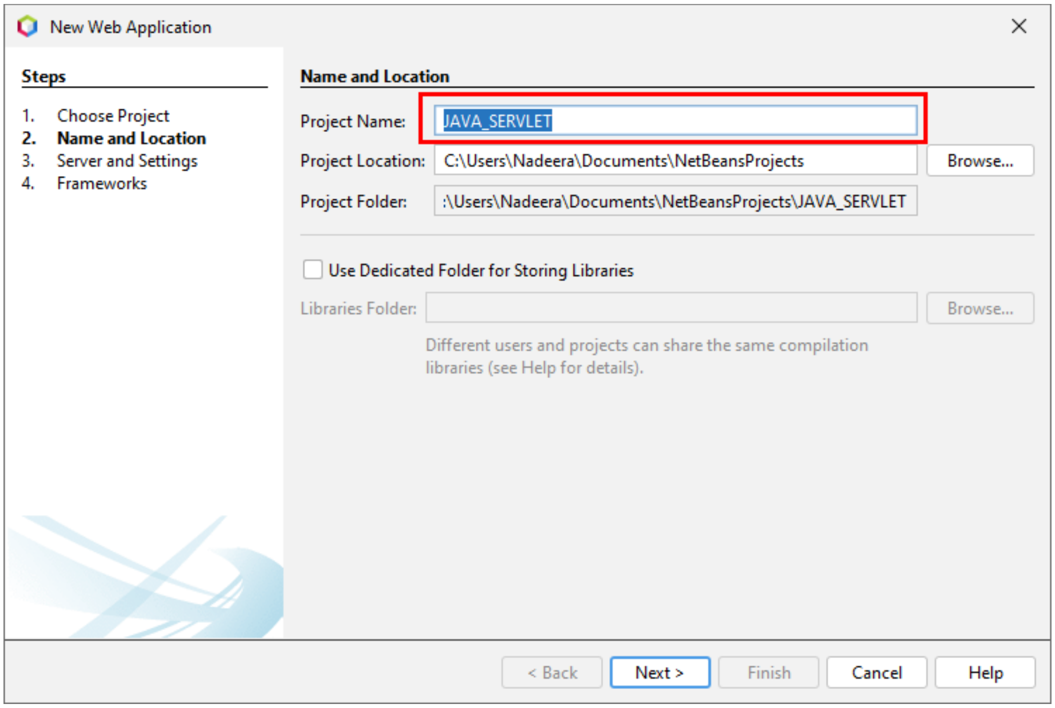
Create Java Servlet Project using NetBeans

Open NetBeans IDE then Click the “File” menu and select “New Project”. Select the Category Java with Ant -> Java Web -> Web Application



Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

Insert the Project Name



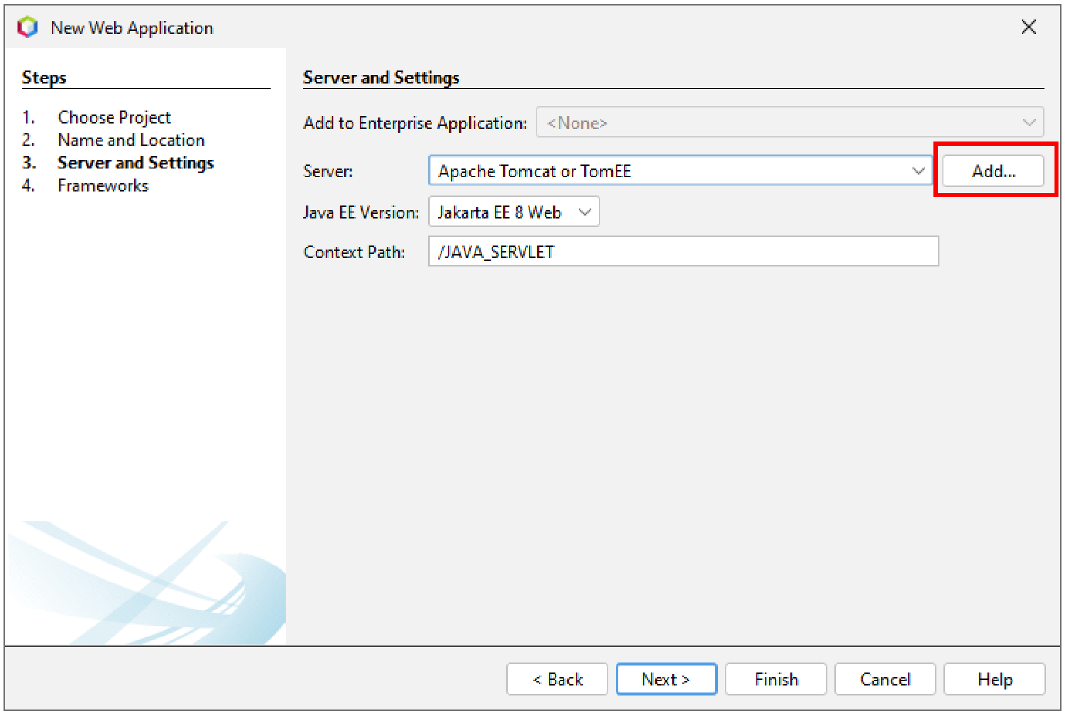
Project Name: JAVA_SERVLET

Project Location: C:\Users\Nadeera\Documents\NetBeansProjects

Project Folder: :\Users\Nadeera\Documents\NetBeansProjects\JAVA_SERVLET

Different users and projects can share the same compilation libraries (see Help for details).

Please Select the Server from the List and Click the Next button. Please click the "Add" button if a server isn't already mentioned.



New Web Application

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

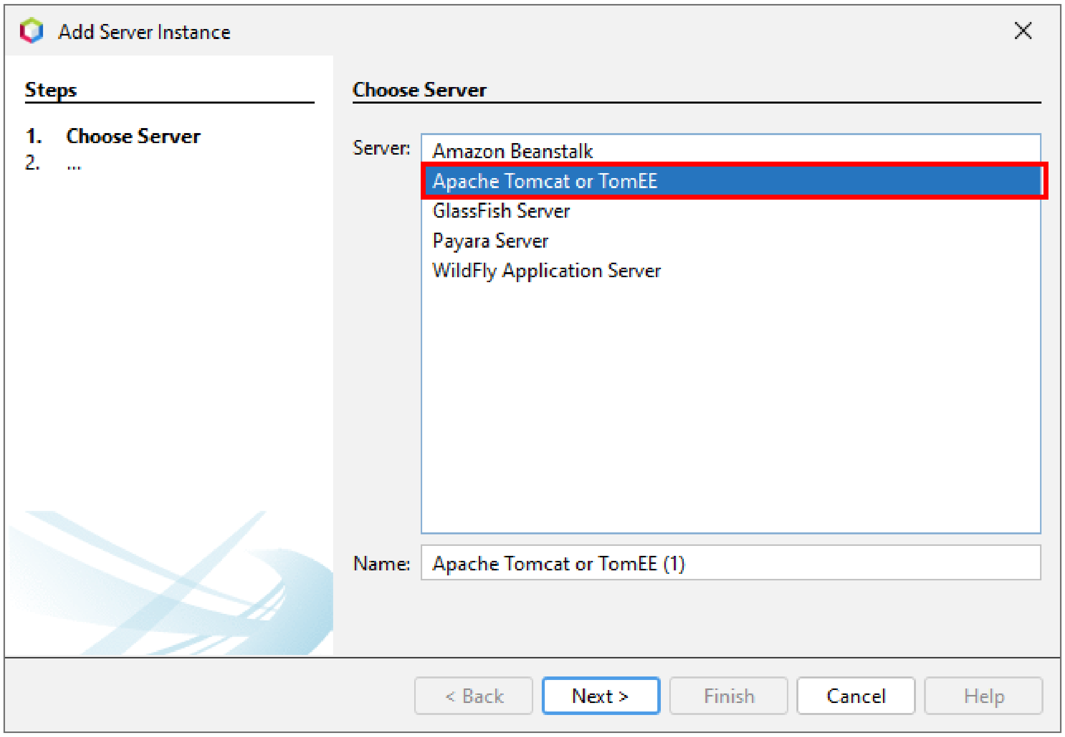
Add to Enterprise Application: <None>

Server: Apache Tomcat or TomEE

Java EE Version: Jakarta EE 8 Web

Context Path: /JAVA_SERVLET

Select Apache Tomcat from the Servers List



Add Server Instance

Steps

1. **Choose Server**
2. ...

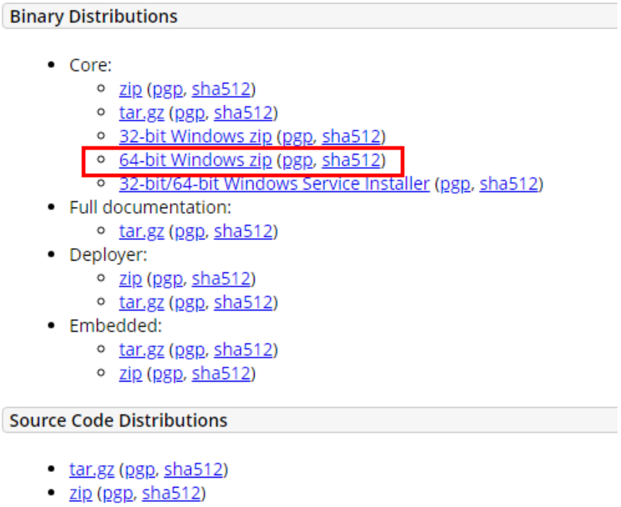
Choose Server

Server:

Amazon Beanstalk
Apache Tomcat or TomEE
GlassFish Server
Payara Server
WildFly Application Server

Name: Apache Tomcat or TomEE (1)

Download Apache Tomcat via https://tomcat.apache.org/download-10.cgi



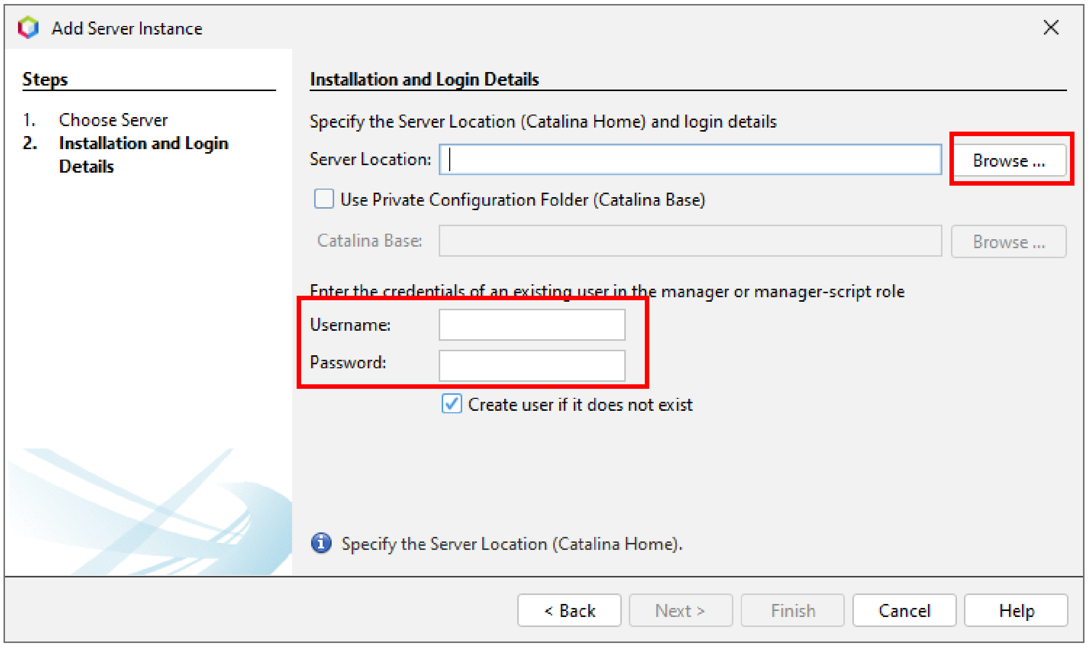
Binary Distributions

- Core:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
  - 32-bit Windows zip (pgp, sha512)
  - 64-bit Windows zip (pgp, sha512)
  - 32-bit/64-bit Windows Service Installer (pgp, sha512)
- Full documentation:
  - tar.gz (pgp, sha512)
- Deployer:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
- Embedded:
  - tar.gz (pgp, sha512)
  - zip (pgp, sha512)

Source Code Distributions

- tar.gz (pgp, sha512)
- zip (pgp, sha512)

Unzip the downloaded file and Browse the extracted folder. Insert a Username and the Password for accessing the admin controls.



Add Server Instance

Steps

1. Choose Server
2. **Installation and Login Details**

Installation and Login Details

Specify the Server Location (Catalina Home) and login details

Server Location:

□ Use Private Configuration Folder (Catalina Base)

Catalina Base:

Enter the credentials of an existing user in the manager or manager-script role

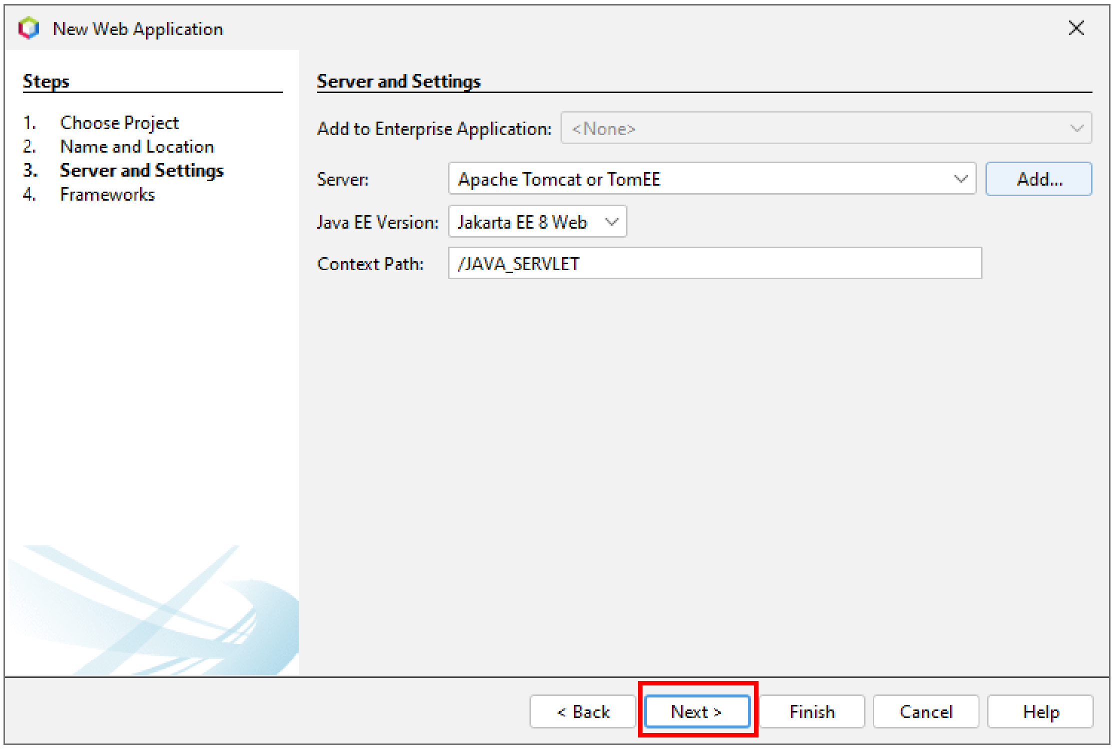
Username:

Password:

☑ Create user if it does not exist

Specify the Server Location (Catalina Home).

Select the Server from Drop down List and Click the Next Button.



New Web Application

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

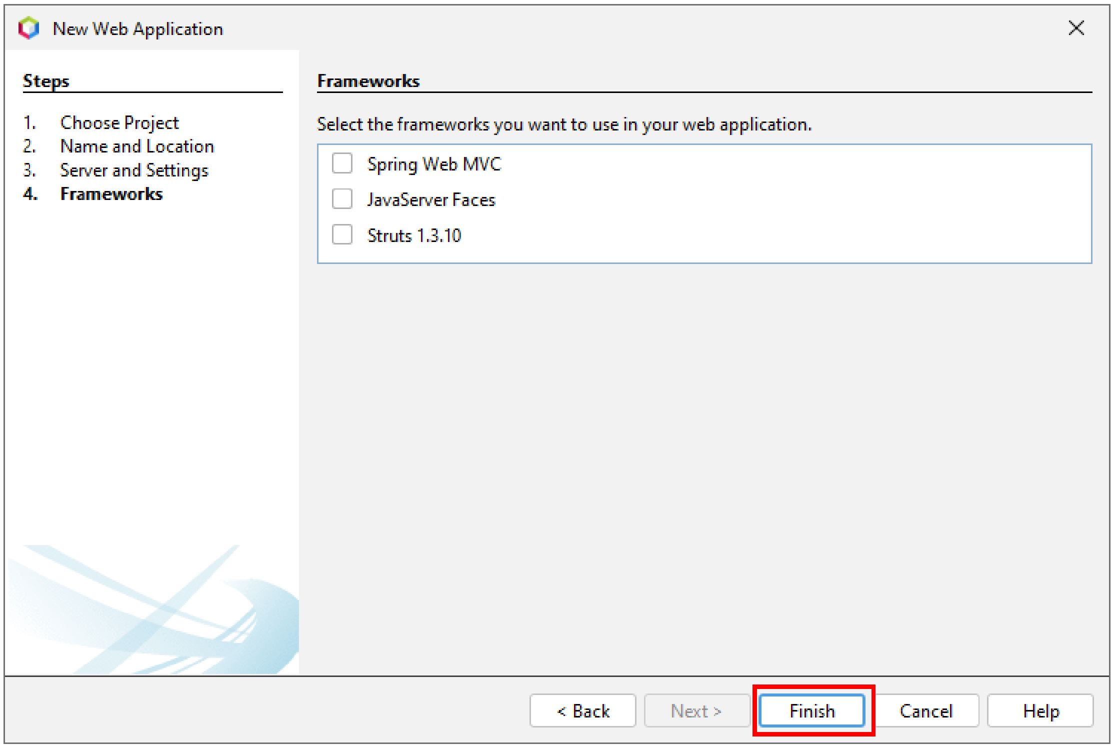
Add to Enterprise Application: <None>

Server: Apache Tomcat or TomEE

Java EE Version: Jakarta EE 8 Web

Context Path: /JAVA_SERVLET

Click the Finish Button to Create the Project.



New Web Application

Steps

1. Choose Project
2. Name and Location
3. Server and Settings
4. **Frameworks**

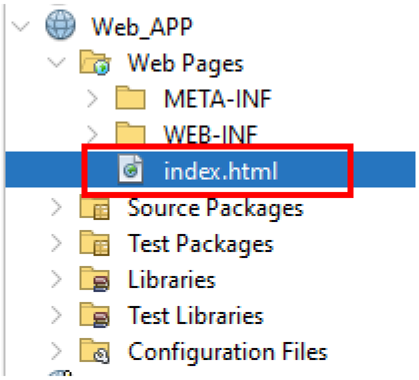
Frameworks

Select the frameworks you want to use in your web application.

☐ Spring Web MVC
☐ JavaServer Faces
☐ Struts 1.3.10

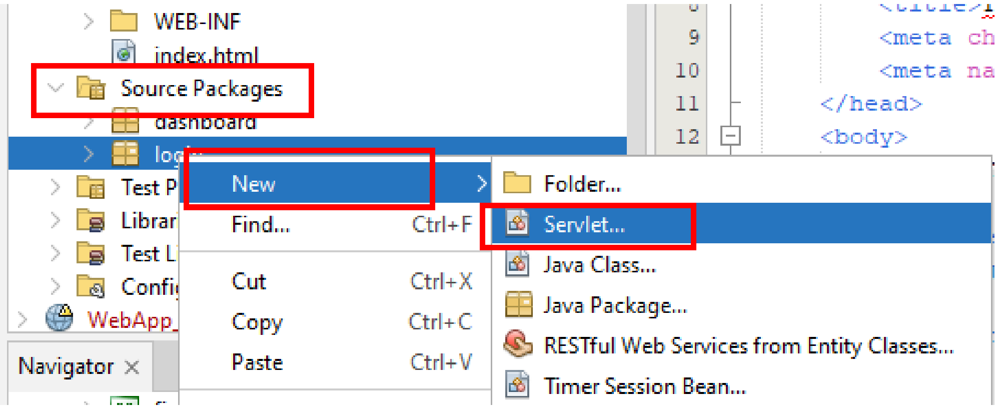
GET and POST Requests

In the index.html file, create two HTML forms.



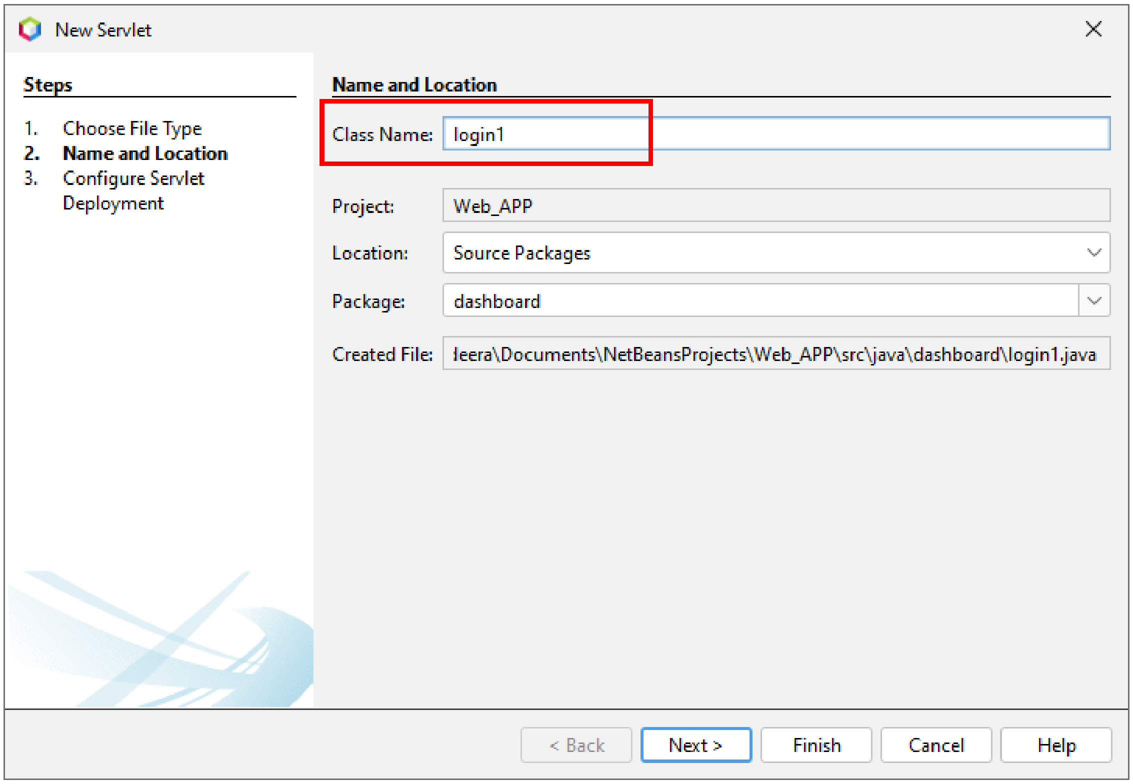
```
<div>GET AND POST METHODS</div>

    <fieldset>
      <legend>GET</legend>
      <form name="login" action="login" method="GET">
        <label>First Name</label>
        <input type="text" name="fname"/>
        <br><br>
        <label>Last Name</label>
        <input type="text" name="lname"/>
        <br><br>
        <input type="submit" value="Submit"/>
      </form>
    </fieldset>
    <br><br>
    <fieldset>
      <legend>POST</legend>
      <form name="login" action="login" method="post">
        <label>UserName</label>
        <input type="text" name="uname"/>
        <br><br>
        <label>Password</label>
        <input type="password" name="pass"/>
        <br><br>
        <input type="submit" value="Login"/>
      </form>

    </fieldset>
```

Create “login” Servlet Page. The Servlet Page name needs to be added to the appropriate HTML form's action attribute.



```
 9          <meta ch
10          <meta na
11      </head>
12      <body>
```

Insert the Servlet Page name and Click next Button



New Servlet

Steps

1. Choose File Type
2. **Name and Location**
3. Configure Servlet Deployment

Name and Location

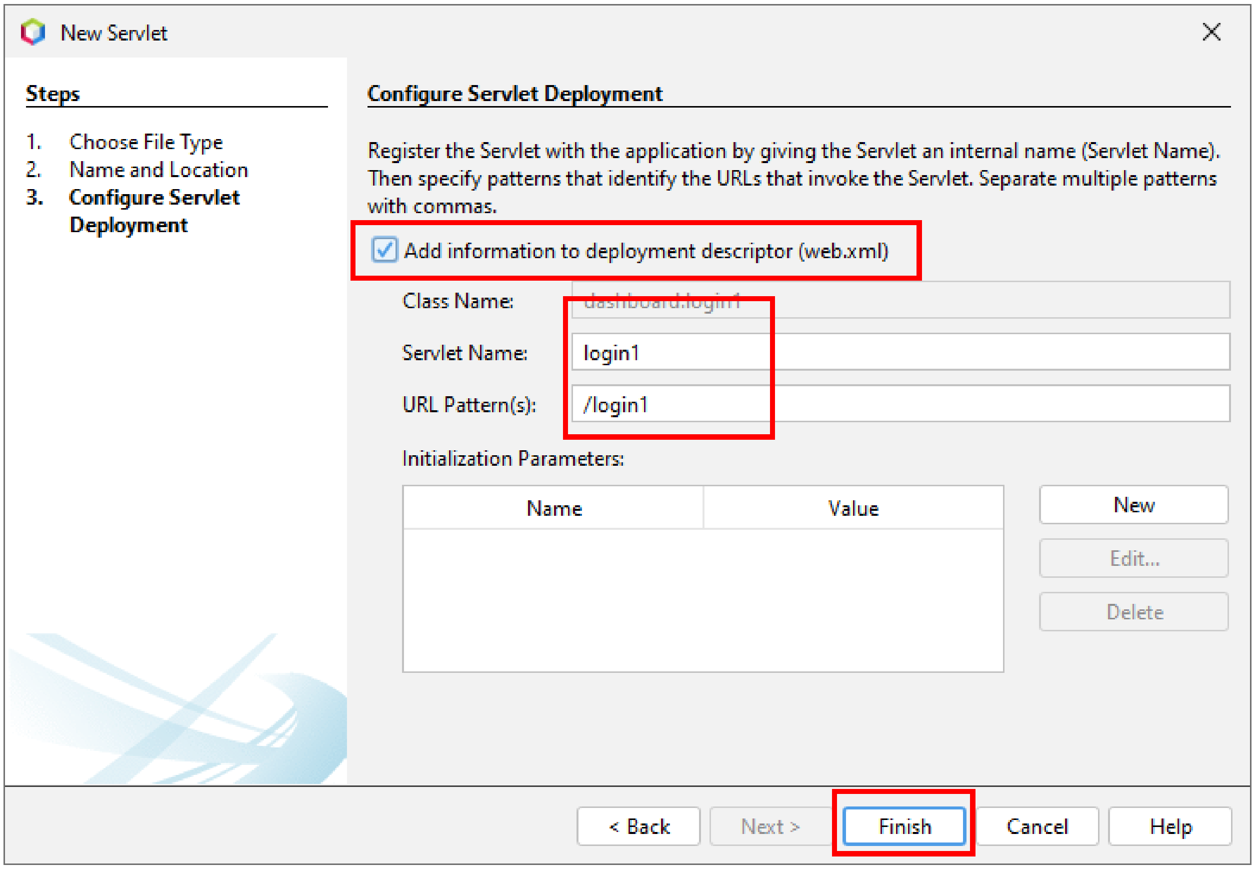
Class Name: login1

Project: Web_APP

Location: Source Packages

Package: dashboard

Created File: Ieera\Documents\NetBeansProjects\Web_APP\src\java\dashboard\login1.java



Configure Servlet Deployment

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

☑ Add information to deployment descriptor (web.xml)

Class Name: dashboard.login1

Servlet Name: login1

URL Pattern(s): /login1

Initialization Parameters:

Name	Value

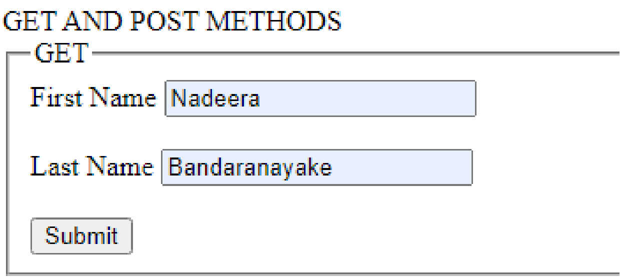
Update **doGet** and **doPost** Methods in Created Servlet Page File

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String fname = request.getParameter("fname");
    String lname = request.getParameter("lname");
    String full_Name = fname + " " + lname;

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet login</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h4>First Name " + fname + "</h4>");
        out.println("<h4>Last Name " + lname + "</h4>");
        out.println("<h4>Full Name " + full_Name + "</h4>");
        out.println("</body>");
        out.println("</html>");
    }
  }
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String uname = request.getParameter("uname");
    String pass = request.getParameter("pass");

    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet login</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h4>UserName " + uname + "</h4>");
        out.println("<h4>Password " + pass + "</h4>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

getParameter() – You call request.getParameter() method to get the value of a form parameter.

Run your Project and Click the Submit Button.



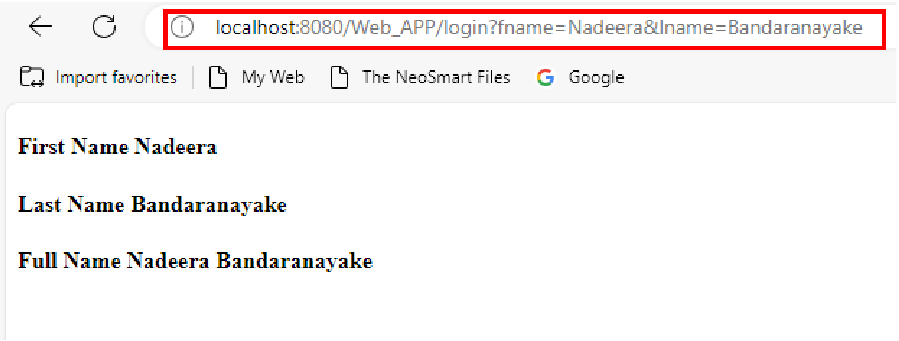
GET AND POST METHODS

GET

First Name Nadeera

Last Name Bandaranayake

View the result



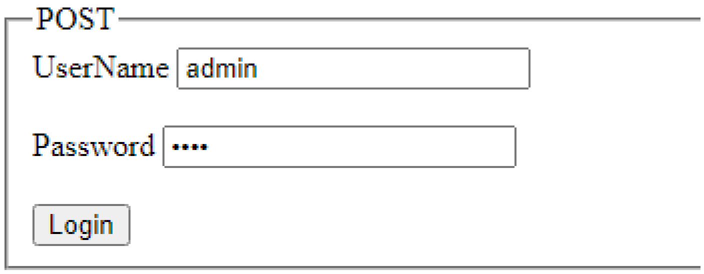
localhost:8080/Web_APP/login?fname=Nadeera&lname=Bandaranayake

First Name Nadeera

Last Name Bandaranayake

Full Name Nadeera Bandaranayake

Run your Project and Click the Login Button.

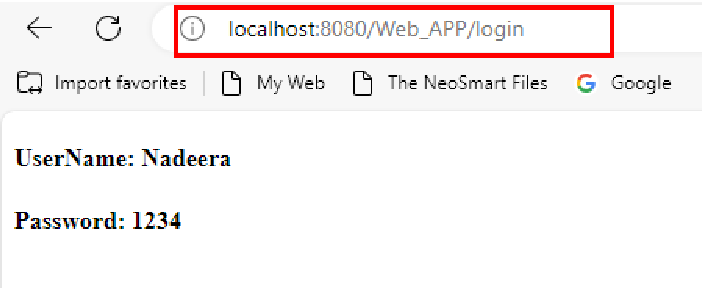


POST

UserName admin

Password ••••

View the result



localhost:8080/Web_APP/login

UserName: Nadeera

Password: 1234

Servlet with JDBC

Create a Login Page. Ex. login.html

```
<html>
  <head>
    <title>JAVA Servlet</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <form name="login" action="login" method="post">
      <label>UserName</label>
      <input type="text" name="uname"/>
      <br><br>
      <label>Password</label>
      <input type="password" name="pass"/>
      <br><br>
      <input type="submit" value="Login"/>
    </form>
  </body>
</html>
```

Crea a MySQL Database and Table

Database Name: java_test

Table Name: users

```
CREATE TABLE `users` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `name` varchar(20) NOT NULL,
 `email` varchar(30) NOT NULL,
 `password` varchar(20) NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;
```

Create a “DBConnection” Java class to connect MySQL databases.

The dbCon method is responsible for establishing a connection with the MySQL database and returning the connection.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
  private static Connection con;

  public static Connection dbCon(){

    try{
      final String connectionUrl = "jdbc:mysql://localhost:3306/java_test?serverTimezone=UTC";
      final String user="nadeera";
      final String password="Admin@1234";
      con =  DriverManager.getConnection(connectionUrl, user, password);
    }
    catch(SQLException ex){
      System.out.println("Error"+ex);
    }

    return con;
  }
}
```

Create a “Validate” Java class to validate users.

The LoginValidate function obtain the password and username as parameters, then verify them in the Users table. The method, Returns true if the username and password are in the table; returns false otherwise.

```
import java.sql.*;

public class Validate {
    public static boolean loginValidate(String username, String pass) {

        boolean valid= false;

        try{

             //Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DBConnection.dbCon();
            Statement stmt = conn.createStatement();
            String query="SELECT * FROM users WHERE name='"+username+"' AND password='"+pass+"'";
            ResultSet rs = stmt.executeQuery(query);

            valid = rs.next();
        }
        catch(SQLException e){//| ClassNotFoundException
            System.out.println("Error: "+e);
        }

        return valid;
    }
}
```

“Login” Servlet Page

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String uname = request.getParameter("uname");
        String pass = request.getParameter("pass");

        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet login</title>");
        out.println("</head>");
        out.println("<body>");

        if (Validate.loginValidate(uname, pass)) {
            response.sendRedirect("welcome");
        } else {
            out.print("Login Failed");
        }
        out.println("</body>");
        out.println("</html>");

    }
```

The doPost method get the username and password from the login.html form and send them to the login validate method. If the login validate method returns true this will redirect to the welcome page otherwise it print Login Failed.

Sessions and Cookies in Servlet

In Servlets, sessions and cookies are mechanisms used to maintain state information (current condition) and manage user interactions on a web application.

Session

A session refers to a logical connection or interaction between a web client (typically a browser) and a web server over a period of time. It allows the server to maintain stateful information about a series of related requests and responses from the same client. Session is used to save user information on the server.

- A session represents a series of related HTTP requests and responses between a client and a server within a specific timeframe.
- Sessions are used to store user-specific data across multiple requests, allowing web applications to maintain stateful interactions with clients.
- In Servlets, sessions are managed using the HttpSession interface provided by the Servlet API.
- Data stored in a session is accessible to all servlets within the same application context and is maintained on the server side.

Cookie

Cookies are the textual information that is stored in key-value pair format to the client's browser during multiple requests.

- Cookies are small pieces of data sent from a web server and stored on the client's browser.
- Cookies are commonly used to store user preferences, session identifiers, authentication tokens, and other client-specific information.
- In Servlets, cookies are managed using the Cookie class provided by the Servlet API.
- Cookies are sent with each HTTP request to the server, allowing the server to read and manipulate them.

- Cookies have limitations such as size restrictions and security concerns, so sensitive information should not be stored directly in cookies.

Update the Login Page with Cookies and Sessions

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();

    String uname = request.getParameter("uname");
    String pass = request.getParameter("pass");

    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet login</title>");
    out.println("</head>");
    out.println("<body>");

    if (Validate.loginValidate(uname, pass)) {

        //Create Session
        HttpSession session = request.getSession();
        session.setAttribute("username", uname);

        //Setting a Cookie
        Cookie ck = new Cookie("cname", uname);
        ck.setMaxAge(60 * 60);
        response.addCookie(ck);

        response.sendRedirect("welcome");
    } else {
        out.print("Login Failed");
    }

    out.println("</body>");
    out.println("</html>");

  }
```

Create a session object if it is already not created.

HttpSession session = request.getSession();

store the user information into the session object - setAttribute(String name, Object value)

session.setAttribute("username", uname);

Creating a Cookie object - Cookie cookie = new Cookie("key","value");
Cookie ck = new Cookie("cname", uname);

Setting the maximum age − You use setMaxAge to specify how long (in seconds) the cookie should be valid.
ck.setMaxAge(60 * 60);

Sending the Cookie into the HTTP response headers
response.addCookie(ck);

Update the Welcome Page

Processes requests method used for both HTTP GET and POST methods

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();

    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet welcome</title>");
    out.println("</head>");

    HttpSession session = request.getSession(false);
    //Check the Session
    if (session == null || session.getAttribute("username") == null) {
      response.sendRedirect("login.html");
    } else {

      //get cname from cookie
      Cookie cookies[] = request.getCookies();
      if (cookies != null) {
        String name = cookies[0].getValue();
        for (int i = 0; i < cookies.length; i++) {
          if ("cname".equals(cookies[i].getName())) {
            name = cookies[i].getValue();
          }
        }
        out.println("<h4>Hi," + name + "</h4>");
      }
      out.println("<body>");
      out.println("<h1>Dashboard</h1>");
      out.println("<a href='Logout'>Log out</a>");
    }
    out.println("</body>");
    out.println("</html>");
  }
```

The page loads the current HttpSession associated with this request, and if there is no current session or the session username is null, it will redirect to the login page.

Also, it retrieves the value of the "cname" cookie from the cookies and prints its value on the webpage.

Create a Logout Page

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        request.getRequestDispatcher("login.html").include(request, response);


        HttpSession session=request.getSession();
        session.invalidate();

        out.print("You are successfully logged out!");

        out.close();
    }
```

Once the user requests to logout, we need to destroy that session. To do this, we need to use invalidate() method in HttpSession Interface.

After the session is destroyed, the page will redirect to the login.html page.

RequestDispatcher is an interface provided by the Servlet API. It allows servlets to forward requests or include responses to other servlets essentially, it helps in dispatching the client's request from one servlet to another or including content from another resource into the response.

ENTERPRISE ARCHITECTURE

- LECTURE: M.F.M.SAMLY
- EMAIL: M.SHAMLYMHMD@GMAIL.COM
- COURSE CODE: HNDIT4232
- COURSE STATUS: COMPULSORY, GPA
- COURSE COORDINATOR: HOD

Course Introduction

- Principle of distributed software development.
- Working with database.
- Web base application development and state management.
- XML data representation and processing.
- Servlets Overview.
- Development of servlets.
- Java server pages.
- Applications of EJB.
- STRUTS
- Hibernate
- Spring

How you will be evaluated…

Online Quiz & Group Assignment	**40%**
6 Assignments	
Final Examination	**60%**
Total	**100%**

PRINCIPLES OF DISTRIBUTED SOFTWARE DEVELOPMENT

1. Introduction to Distributed Software Development
2. Role of Java in Enterprise Development
3. Java in Enterprise Development
4. Enterprise Application Architecture

1. Introduction to Distributed Software Development

Distributed software development involves creating applications that run on multiple interconnected computers.

- **Modularity**: Break down the system into manageable modules
- **Loose Coupling**: Reduce dependencies between modules
- **Scalability**: Ability to handle increasing workloads
- **Reliability**: System remains operational under varying conditions
- **Security**: Protect data and resources from unauthorized access

What is distributed application/system?

A distributed application is a software application that consists of multiple components running on different computers connected by a network.

- Web applications
- Multiplayer online games
- Distributed databases
- Internet of Things (IoT) applications

2. Role of Java in Enterprise Development

Java provides libraries and frameworks (e.g., Apache Kafka) for building distributed systems.

- **Platform Independence**: Write once, run anywhere
- **Robustness**: Strong memory management, garbage collection
- **Security**: Built-in security features
- **Scalability**: Ability to handle large-scale applications
- **Performance**: Optimized virtual machine (JVM)

3. Java in Enterprise Development

1. **Web Applications**: Servlets, JSP, Spring Framework
2. **Enterprise Integration**: JMS, JNDI, EJB
3. **Database Connectivity**: JDBC, Hibernate
4. **Messaging**: JMS (Java Message Service)
5. **Web Services**: SOAP, RESTful services

4. Enterprise Application Architecture

- Components

- **Presentation Layer**: User interface (UI)
- **Business Logic Layer**: Application logic
- **Data Access Layer**: Database interactions
- **Integration Layer**: Connects with external systems

- Java Technologies

- **Servlets and JSP**: Presentation layer
- **Spring Framework**: Business logic and integration
- **Hibernate**: Data access layer
- **JMS**: Messaging between components

Example: Online Retail System (Web Site)

Components

- **Presentation Layer**: Web interface for customers to browse and purchase products.
- **Business Logic Layer**: Manages shopping cart, order processing, and inventory management.
- **Data Access Layer**: Connects to database to store and retrieve product and customer information.
- **Integration Layer**: Interfaces with payment gateways, shipping services, and external systems.

Java Technologies Used

- **Servlets and JSP**: Presentation layer, handling user interactions and displaying content.
- **Spring Framework**: Business logic, providing services such as shopping cart management and order processing.
- **Hibernate**: Data access layer, interacting with the database to store and retrieve product and customer data.
- **JMS**: Messaging between components for asynchronous processing and communication with external systems.

SLIATE - Kegalle

Q & A

THANK YOU

WORKING WITH DATABASE

Database

What is database?

A database is a structured collection of data that is organized in such a way that it can be easily accessed, managed, and updated. It typically stores data in tables, with each table containing rows and columns that represent individual records and attributes, respectively.

1. **Relational Databases**: These databases organize data into tables with rows and columns, where each row represents a record and each column represents an attribute. Examples include MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.
2. **NoSQL Databases**: These databases are designed to store and manage unstructured, semi-structured, or structured data. They are often used for big data and real-time web applications. Examples include MongoDB, Cassandra, and Redis.

Software Packages And Tools

1. **MySQL Workbench**
2. **Microsoft SQL Server Management Studio (SSMS)**
3. **Oracle SQL Developer**
4. **SQLite**
5. **PostgreSQL**
6. **MongoDB Compass**
7. **DBeaver**

JDBC

JDBC (Java Database Connectivity) is an API that enables Java applications to interact with databases.

- **JDBC Architecture**
  - **Application**: The Java application that interacts with the database.
  - **JDBC API**: Defines the interfaces and classes for JDBC programming.
  - **Driver Manager**: Manages the JDBC drivers and provides methods for establishing connections.
  - **JDBC Driver**: The driver implements the JDBC interfaces to communicate with the database.

JDBC Connection Code

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class MySQLExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/DATABASE";
        String username = “USERNAME";
        String password = "PASSWORD";
        try {
            Connection conn = DriverManager.getConnection(url, username, password);
            System.out.println("Connected to MySQL database!");
            conn.close();
        } catch (SQLException e) {
            System.out.println("Failed to connect to MySQL database!");
            e.printStackTrace();
        }
    }
}
```

Try - Catch

In Java, the try and catch blocks are used for exception handling.

Exceptions are events that occur during the execution of a program that disrupt the normal flow of instructions. They can occur for various reasons, such as invalid input, file not found, or database connection issues.

Can you write query for this?

1. Can you create a database call “System” and create a table call “user”?

   Columns are Id (should be unique), Name, Email, Phone number and address

2. Can you design a data entering management system above column?

3. Can you connect system with database?

(If user entered, data should be store in database.)

4. Can you design a table and retrieve all the data (using array list with get set method) what you stored?

DBMS vs RDBMS

- **DBMS**

Data is stored in a database management system (DBMS) as a file

Data is stored in a database management system (DBMS) in either a navigational or hierarchical format

Only a single user is supported by the DBMS

- **RDBMS**

Tables are used to store information

RDBMS employs a tabular format, with column names as headers and associated data as rows

It may be used by numerous people

DBMS is an acronym for

- (A) Database Merging System
- (B) Database
- (C) Database Management System
- (D) Database Manipulating System

RDBMS is an acronym for

- (A) Relational Database
- (B) Relational Database Merging System
- (C) Relational Database Management System
- (D) Relational Database Manipulation System

Q & A

Thank You!

Web-based Application Development And State Management

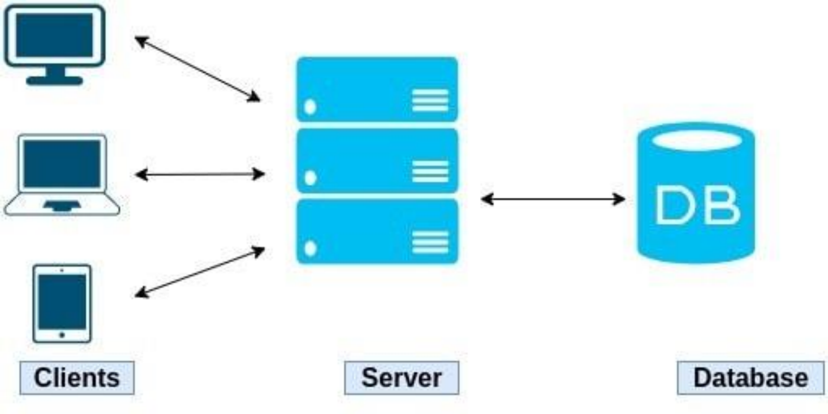
3-TIER APPLICATION MODE

What is web-based Application?

- Web-based applications are software programs accessed through web browsers, providing functionality and services over the internet.
- Examples include email clients, social media platforms, online shopping portals, and productivity tools.
- Client-side vs. Server-side Development

Client-side vs. Server-side Development

- Client-side development involves programming executed on the user's device (browser), such as HTML, CSS, and JavaScript for creating interactive interfaces.
- Server-side development involves programming executed on the web server, managing data, business logic, and communication with databases.

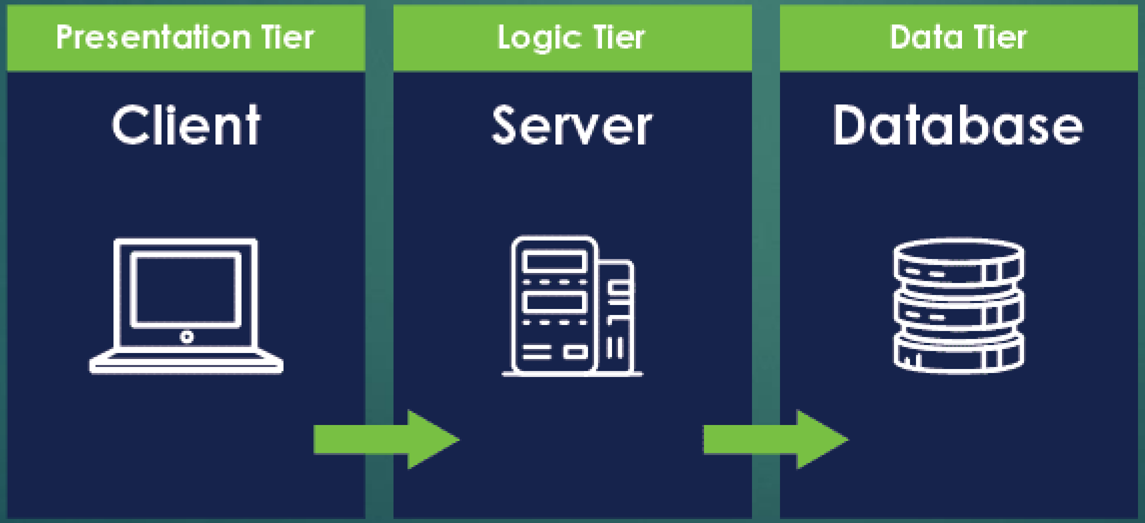


State Management in Web Applications

- **What is State in Web Applications?**
  - State refers to the current condition or data of a web application, including user inputs, session details, and application state (e.g., logged-in status, shopping cart contents).
- **Types of State Management**
  - **Client-Side State Management:**
    - Involves storing and managing state data on the client's browser.
    - Techniques include cookies, local storage, session storage, and client-side frameworks/libraries (e.g., Redux for React).
  - **Server-Side State Management:**
    - Involves storing state data on the server, often in databases or server memory.
    - Techniques include sessions, databases, and server-side frameworks/libraries (e.g., Express.js for Node.js).

3-Tier Application Model

- Presentation Layer (UI Layer)
- Business Logic Layer (Middle Tier)
- Data Layer (Backend)



3-Tier Application Model

- Presentation Layer (UI Layer)
  - **Components of the Presentation Layer**
    - **Web Pages:** HTML documents that structure content and layout.
    - **JavaScript:** Enhances interactivity, dynamic behavior, and client-side validation.
    - **User Interface Elements:** Buttons, forms, menus, navigation bars, modals, and interactive elements.
  - The Presentation Layer plays a crucial role in delivering a visually appealing, interactive, and user-friendly experience in web applications.

3-Tier Application Model

- Business Logic Layer (Middle Tier)
  - Implements core application logic, algorithms, and business rules.
  - Acts as a bridge between the Presentation Layer (UI) and the Data Layer (Backend), ensuring data integrity and security.
  - **Data Validation:** Validates user inputs, form submissions, and data integrity checks (e.g., email validation, password strength).
  - **Authorization and Access Control:** Enforces user roles, permissions, and access control policies (e.g., user authentication, role-based access).

- Example: Consider an e-commerce platform's Business Logic Layer handling order processing, inventory management, pricing calculations, and user authentication.

3-Tier Application Model

- Data Layer (Backend)
  - Manages data storage, retrieval, and manipulation operations.
  - Ensures data integrity, security, and scalability.
  - Provides interfaces for interacting with databases, file systems, external APIs, and data sources.

What are Dynamic Web Models?

- Dynamic web models refer to architectures and patterns that facilitate dynamic content generation, user interactions, and real-time updates in web applications.
- These models enhance user experience, data presentation, and responsiveness by leveraging technologies like AJAX, WebSocket, and server-side rendering.

- **AJAX (Asynchronous JavaScript and XML)**
  - Allows asynchronous data retrieval and updating parts of a web page without full page reloads.
- **WebSockets**
  - Enables full-duplex communication between the client and server, facilitating real-time data exchange and event-driven interactions.
- **Single Page Applications (SPAs)**
  - Loads a single HTML page and dynamically updates content using JavaScript frameworks/libraries (e.g., React, Angular, Vue.js).

Benefits Of Dynamic Web Models?

- **Improved User Experience:** Real-time updates, interactive features, and seamless navigation enhance user satisfaction.
- **Efficient Data Handling:** Asynchronous data fetching, live updates, and optimized rendering improve application performance.
- **Scalability:** Dynamic web models support scalability, allowing applications to handle increased user traffic and data volumes.

MVC Architecture

- MVC is a design pattern that divides an application into three interconnected components
  - **Model:** Represents the data and business logic of the application. Manages data manipulation, validation, and interactions with the database.
  - **View:** Represents the user interface (UI) components responsible for presenting data to users. Displays information and interacts with users through forms, buttons, etc.
  - **Controller:** Acts as an intermediary between the Model and View. Handles user input, processes requests, invokes business logic in the Model, and updates the View.

MVVC Architecture

- MVVM is a design pattern that divides an application into three main components.
  - **Model:** Represents the data and business logic of the application, similar to MVC.
  - **View:** Represents the user interface (UI) elements visible to users, similar to MVC.
  - **ViewModel:** Acts as an intermediary between the View and Model, providing data binding, UI logic, and state management.

What is State Management?

- State refers to the current condition or data of an application, including user inputs, session details, and application state.
- Effective state management ensures data persistence, synchronization, and seamless user experience across pages and sessions.
- **State Management**
  - **Client-Side State Management**
  - **Server-Side State Management**

Client-Side State Management

- Involves storing and managing state data on the client's browser.
- Techniques include cookies, local storage, session storage, and client-side frameworks/libraries (e.g., Redux, Vuex).
- **Client-Side State Management Techniques**
  - **Cookies:**
    - Small text files stored on the client's browser, used for storing user preferences, session identifiers, and tracking information.
    - Limited storage capacity and security considerations (e.g., privacy, tampering).
  - **Local Storage and Session Storage:**
    - HTML5 APIs for storing key-value pairs on the client side.
    - Local storage persists data across browser sessions, while session storage is cleared when the session ends (e.g., browser/tab closure).
  - **Client-Side Frameworks/Libraries:**
    - Redux (React), Vuex (Vue.js), NgRx (Angular) for centralized state management in frontend applications.
    - Provide predictable state updates, middleware for asynchronous actions, and dev tools for debugging.

Server-Side State Management

- Involves storing state data on the server, often in databases or server memory.
- Techniques include sessions, databases, and server-side frameworks/libraries (e.g., Express.js middleware, Spring Session).
- **Server-Side State Management Techniques**
  - **Sessions:**
    - Server-side storage of session data associated with a user's session ID (e.g., login credentials, shopping cart items).
    - Maintained across multiple requests until the session expires or is invalidated.
  - **Databases:**
    - Persistent storage of application state and user data in relational or NoSQL databases (e.g., MySQL, MongoDB).
    - Offers scalability, reliability, and data integrity for managing complex state information.
  - **Server-Side Frameworks/Libraries:**
    - Express.js middleware, Spring Session for handling session management and state storage on the server side.

Q & A

Thank You!

XML

What is XML?

- XML stands for Extensible Markup Language.
- It is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- XML is used to represent hierarchical data structures such as documents, configurations, and data records.

XML Components Overview

- **XML Tags**
- XML tags are used to define elements within an XML document.
- Tags enclose data and represent the structure of the document.
- Example: <book> is a tag that defines a book element.

- **Attributes**
- Attributes provide additional information about elements.
- They are specified within the start tag of an element.
- Example: <book id="101"> has an attribute "id" with the value "101".

- **Elements**
- Elements are the building blocks of an XML document.
- They can contain text, other elements, or a combination of both.
- Example: <title>XML Basics</title> is an element with the name "title".

Example

```
<library>
  <book id="101">
    <title>XML Basics</title>
    <author>John Doe</author>
    <genre>Programming</genre>
    <price>29.99</price>
  </book>
  <book id="102">
    <title>Java Programming</title>
    <author>Jane Smith</author>
    <genre>Programming</genre>
    <price>39.99</price>
  </book>
</library>
```

XML Document Structure

Structure of an XML Document

- XML documents are structured as a hierarchy of elements.
- Each XML document must have a single root element that contains all other elements.

Root Element

- The root element is the top-level element in an XML document.
- It encapsulates all other elements within the document.

Nested Elements

- XML elements can be nested within each other, creating a hierarchical structure.
- Parent elements contain child elements, forming a tree-like structure.

Advantages of Using XML in Enterprise Applications

Data Interchange and Interoperability

- XML enables seamless data interchange between different systems and platforms.
- It facilitates communication and integration between diverse applications and technologies.
- Example: XML-based web services can exchange data between a Java application and a .NET application.

Platform Independence

- XML data is platform-independent, meaning it can be processed and interpreted on various operating systems and devices.
- It promotes compatibility and reduces dependencies on specific technologies.
- Example: An XML document created on a Windows system can be accessed and processed on a Linux system.

Continue…

Extensibility and Flexibility

- XML allows for the creation of custom data structures and schemas tailored to specific requirements.
- It supports hierarchical data representation and complex data models.
- Example: An e-commerce platform can use XML to define product catalogs, orders, and customer information in a structured manner.

Structured Data Representation

- XML provides a structured format for organizing and storing data elements, attributes, and relationships.
- It enhances data clarity, readability, and organization, making it easier to manage and manipulate.
- Example: An XML document representing a customer profile includes structured elements for name, address, contact information, etc.

XML Processing Technologies

- Document Object Model (DOM)
- Simple API for XML (SAX)
- XML Query Languages (XPath, XQuery)
- XML Transformation Technologies (XSLT)

Document Object Model (DOM)

- DOM is an in-memory representation of an XML document as a tree structure.
- It allows programs to dynamically access, modify, and manipulate XML data.
- Example: In Java, DOM parsers like JAXP provide APIs to traverse and manipulate XML documents using DOM.

Simple API for XML (SAX)

- SAX is an event-driven XML parsing model that processes XML sequentially.
- It reads XML data as a stream of events (e.g., start element, end element, character data) rather than loading the entire document into memory.
- Example: SAX parsers in Python, such as xml.sax module, are used for efficient XML parsing in large documents.

XML Query Languages

- XPath: XPath is a language for navigating XML documents and selecting nodes based on criteria.
- XQuery: XQuery is a powerful query language for querying and extracting data from XML documents.
- Example: XPath expressions can be used to locate specific elements or attributes within an XML document.

XML Transformation Technologies

- XSLT (Extensible Stylesheet Language Transformations): XSLT is used for transforming XML documents into different output formats (e.g., HTML, XML, text) using XSL stylesheets.
- Example: XSLT can convert an XML document into an HTML page with styled content for web display.

Quiz

1. What does XML stand for?
   A) Extra Markup Language
   B) Extensible Markup Language
   C) Extended Markup Language
   D) Exclusive Markup Language
2. What is the purpose of XML tags?
   A) To format text in XML documents
   B) To define elements in XML documents
   C) To add comments in XML documents
   D) To hide data in XML documents
3. Which of the following is used to provide additional information about XML elements?
   A) Attributes
   B) Elements
   C) Tags
   D) Values

1. What is the role of the root element in an XML document?
   - A) It contains all other elements in the document
   - B) It provides styling information for the document
   - C) It adds metadata to the document
   - D) It defines the document type
2. Which XML processing technology represents an XML document as a tree structure in memory?
   - A) XQuery
   - B) XPath
   - C) DOM
   - D) SAX
3. What does SAX stand for in XML processing?
   - A) Simple API for XML
   - B) Streamlined Access for XML
   - C) Sequential XML Parser
   - D) Structured API for XML

1. Which XML technology is used for transforming XML documents into different output formats?
   A) XPath
   B) XQuery
   C) XSLT
   D) DOM
2. What is the advantage of using XML in enterprise applications?
   A) Limited compatibility with different platforms
   B) Complex data representation
   C) Seamless data interchange and interoperability
   D) Platform-specific data structures
3. XML allows for the creation of custom data structures and schemas, promoting:
   A) Limited extensibility
   B) Compatibility issues
   C) Platform dependence
   D) Extensibility and flexibility

1. Which XML technology is used for querying and extracting data from XML documents?
   - A) DOM
   - B) XPath
   - C) SAX
   - D) XQuery
2. What is the primary role of XML attributes?
   - A) Define hierarchical structure
   - B) Provide additional information about elements
   - C) Enclose text data
   - D) Define root elements
3. In an XML document, which element encapsulates all other elements?
   - A) Parent element
   - B) Child element
   - C) Root element
   - D) Nested element

Q & A

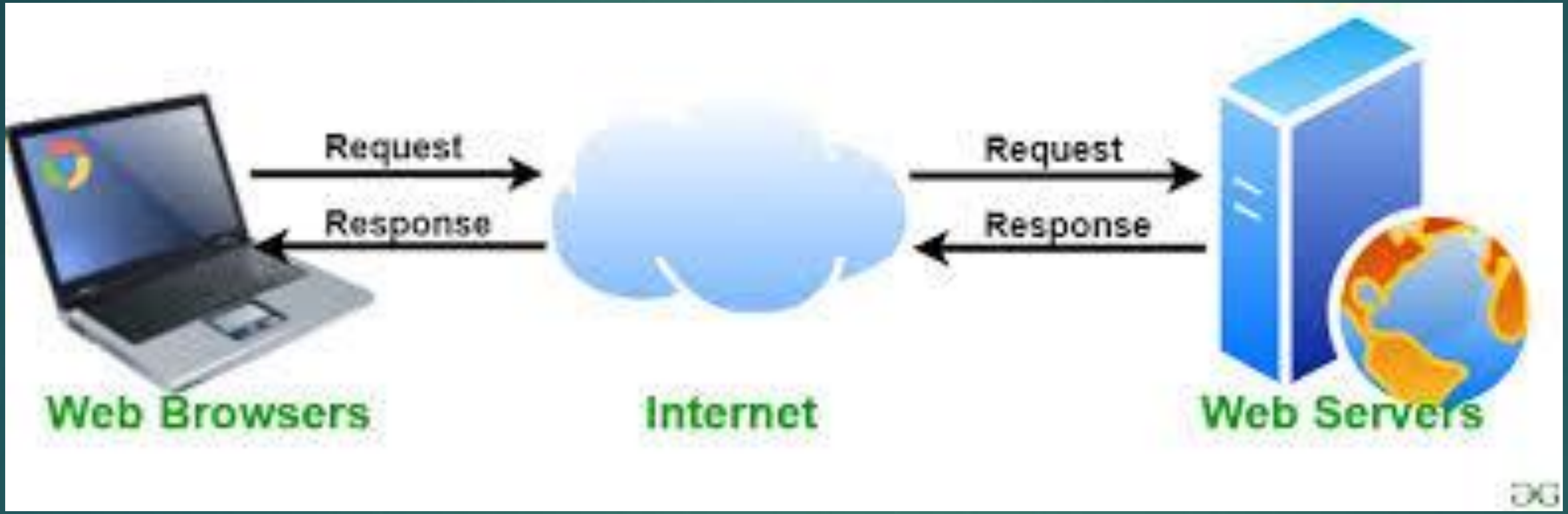
Thank You!

Tools & IDE

WEB SERVERS AND APPLICATION SERVERS

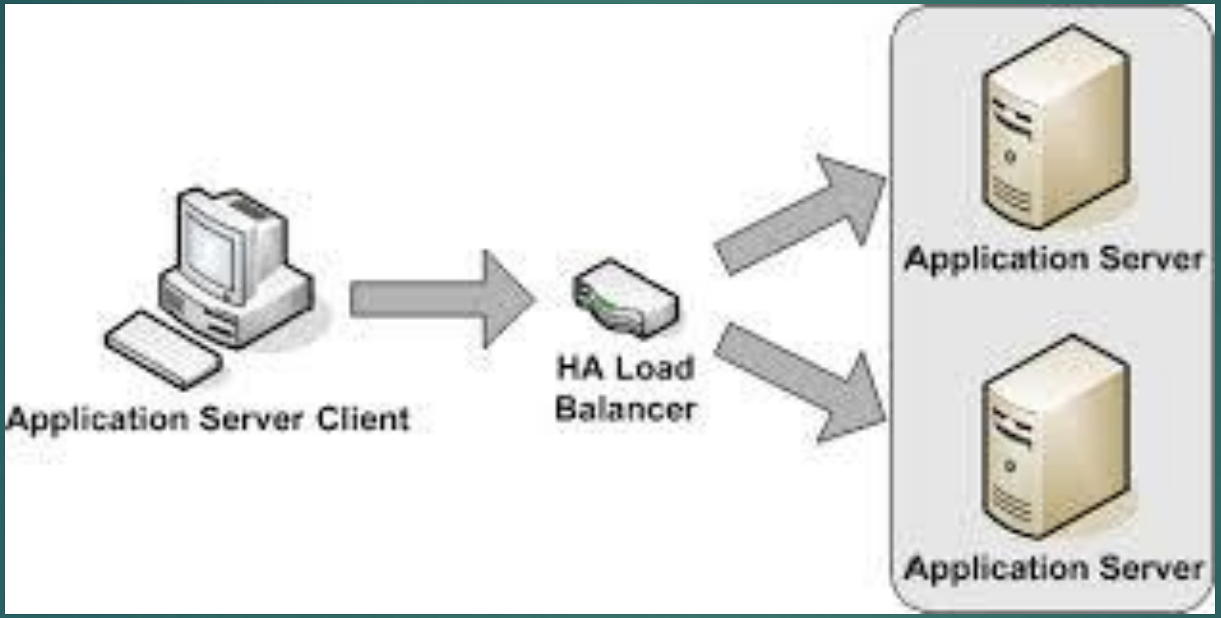
Web Servers

- **Definition:** Web servers are software programs responsible for serving static content to web browsers over the HTTP protocol.
- **Functionality:** They handle requests from clients (browsers) and respond with static HTML, CSS, JavaScript files, images, etc.
- **Examples:** Apache HTTP Server, Nginx, Microsoft IIS.
- **Features:** Load balancing, caching, SSL/TLS support, virtual hosting.
- **Configuration:** Typically configured using configuration files (e.g., httpd.conf for Apache)


Request
Response
Request
Response
Web Browsers
Internet
Web Servers

Application Servers

- **Definition:** Application servers are software frameworks or platforms designed to execute dynamic content and business logic for web applications.
- **Functionality:** They support the execution of server-side code (e.g., Java, PHP, .NET) and provide services such as database access, transaction management, and session handling.
- **Examples:** Tomcat, JBoss (WildFly), IBM WebSphere, Oracle WebLogic.
- **Features:** Java EE (Enterprise Edition) support, clustering, scalability, fault tolerance.
- **Configuration:** Configured through administration consoles or configuration files (e.g., standalone.xml for JBoss).


Application Server Client
HA Load Balancer
Application Server
Application Server

IDE

- An Integrated Development Environment (IDE) is a software suite that consolidates the basic tools developers need to write, test, and debug software.
  - Visual Studio Code
  - NetBeans
  - IntelliJ IDEA
  - Eclipse
- Code Editor
- Compiler/Interpreter
- Debugger
- Version Control
- Build Automation
- Project Management
- Database Integration
- Testing Framework Integration

Eclipse IDE

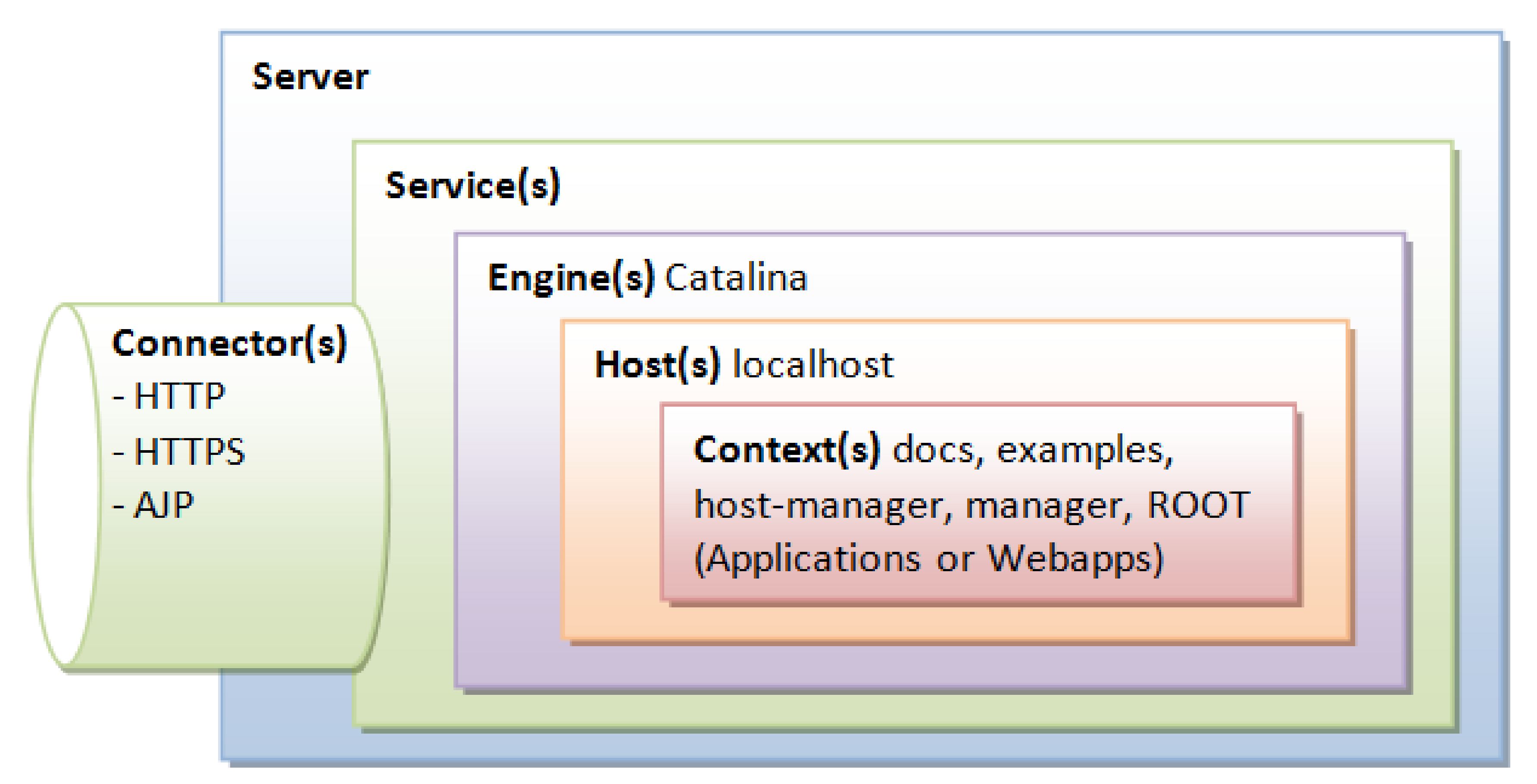
- **Introduction:** Eclipse is a popular integrated development environment for Java development.
- **Features:** Code editing, debugging, version control integration, plugins for web development.
- **Integration:** Eclipse integrates with web servers (e.g., Apache HTTP Server) and application servers (e.g., Tomcat, JBoss) for seamless development and deployment.

Apache Tomcat

- Apache Tomcat is an open-source web server and servlet container developed by the Apache Software Foundation. It provides an environment for running Java-based web applications, specifically servlets and JavaServer Pages (JSP).

1. **Servlet Container:** Tomcat serves as a container for Java servlets, handling requests and responses.
2. **JSP Support:** It supports JavaServer Pages for dynamic web content generation.
3. **Java EE Compatibility:** Tomcat implements Java EE specifications such as Servlet API, JSP API, JDBC API, etc.
4. **HTTP Server:** Acts as a web server capable of handling HTTP requests.

Tomcat Architecture



Server

Service(s)

Engine(s) Catalina

Host(s) localhost

Context(s) docs, examples, host-manager, manager, ROOT (Applications or Webapps)

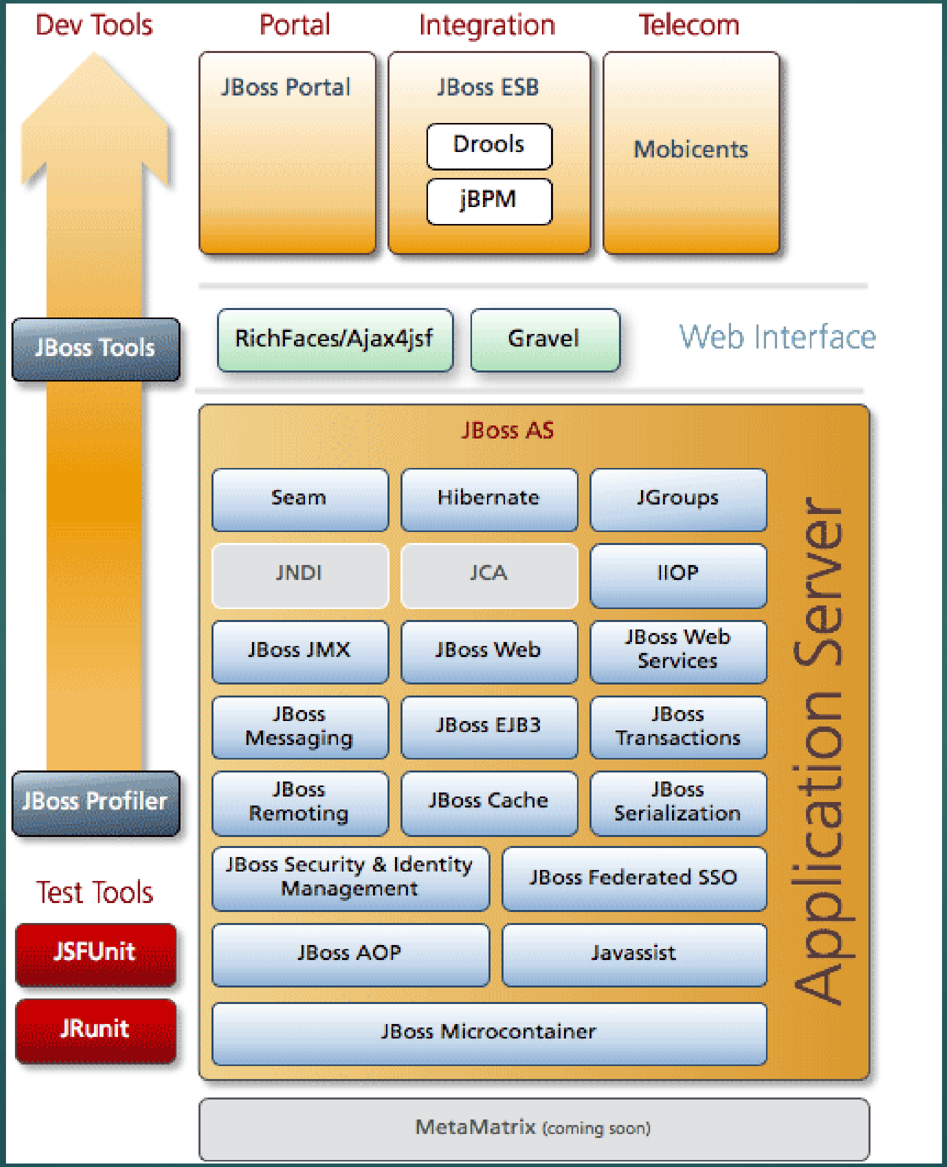
Connector(s)
- HTTP
- HTTPS
- AJP

JBoss Server

- JBoss, now known as WildFly, is an open-source Java EE application server developed by Red Hat.
- It provides a platform for hosting Java-based enterprise applications with support for Java EE specifications.

1. **Java EE Compatibility:** Supports Java EE specifications such as Servlets, JSP, EJB, JPA, CDI, etc.
2. **Modular Architecture:** Modular design for easy customization and scalability.
3. **High Availability:** Clustering and load balancing capabilities for high availability setups.
4. **Management Console:** Web-based administration console for managing server configurations, deployments, and monitoring.

Jboss Architecture



Quiz

- What is the primary function of a web server?
  a) Execute dynamic code ✗
  b) [✓ Serve static content]
  c) Manage databases ✗
  d) Handle client-side scripting ✗
- Which of the following is an example of a web server?
  a) Apache Tomcat
  b) [✓ Microsoft IIS]
  c) IntelliJ IDEA
  d) Eclipse

Quiz

- What does an application server primarily do?
  - a) Serve HTML pages ✗
  - b) Execute client-side scripts ✗
  - [✓ c) Execute server-side code]
  - d) Manage CSS files ✗
- Which of the following is an example of an application server?
  - a) Apache HTTP Server
  - b) Nginx
  - c) Tomcat
  - [✓ d) IBM WebSphere]

Quiz

- Which feature is typically associated with application servers?
  - a) [✓ Load balancing]
  - b) Serving static content
  - c) Managing client requests
  - d) Debugging code
- What is the purpose of an IDE?
  - a) To manage databases ✗
  - b) To serve web pages ✗
  - c) [✓ To write, test, and debug software]
  - d) To handle client requests ✗

Quiz

- Which of the following is NOT an IDE?
  a) Visual Studio Code
  b) NetBeans
  c) [✓ Apache Tomcat]
  d) IntelliJ IDEA
- Which IDE is specifically designed for Java development?
  a) Visual Studio Code ✓
  b) NetBeans
  c) IntelliJ IDEA
  d) [✓ Eclipse]

Quiz

- What is Apache Tomcat primarily used for?
  - a) Writing code
  - b) Serving static content
  - [✓ c) Running Java-based web applications]
  - d) Managing databases
- What type of content does Apache Tomcat handle?
  - a) Static content
  - b) Dynamic content
  - [✓ c) Both static and dynamic content]
  - d) Client-side scripts

Quiz

- What is JBoss Server now known as?
  - a) Apache Tomcat
  - b) [✓ WildFly]
  - c) Apache HTTP Server
  - d) Microsoft IIS
- What does JBoss/WildFly primarily provide for enterprise applications?
  - a) Code editing ✗
  - b) [✓ A platform for hosting Java-based applications]
  - c) Version control integration ✗
  - d) Build automation ✗

Quiz

- Which feature is characteristic of Apache Tomcat?
  - a) Modular architecture
  - b) Clustering and load balancing
  - c) Web-based administration console
  - [✓ d) Java EE compatibility]
- What type of applications does JBoss/WildFly support?
  - a) Only PHP applications
  - b) Only .NET applications
  - [✓ c) Only Java-based enterprise applications]
  - d) Both Java-based and .NET applications

Q & A

Thank You!

Fundamentals of Servlets

SERVLETS OVERVIEW

Introduction to Servlets

What are Servlets?

- Servlets are Java programs that extend the capabilities of a server.
- They handle client requests and generate dynamic responses.
- Part of the Java Enterprise Edition (Java EE) platform.
- Essential for web development in Java.

Role of Servlets

- Serve as the backbone of Java web applications.
- Process HTTP requests and generate HTML, XML, or other formats.
- Enable interaction between clients (browsers) and servers.
- Facilitate dynamic content generation and database connectivity.

Advantages of Using Servlets

1. **Platform Independence:** Servlets run on any server supporting Java.
2. **Robustness:** Built-in exception handling and error management.
3. **Scalability:** Handle multiple requests concurrently.
4. **Session Management:** Maintain stateful communication with clients.
5. **Database Connectivity:** Seamlessly integrate with databases via JDBC.
6. **Security:** Implement authentication, authorization, and data encryption.

Servlet Lifecycle

- **Understanding Servlet Lifecycle Phases**

1. **Initialization Phase**
   - Servlet container loads the Servlet class.
   - init() method is called to initialize the Servlet.
   - Initialization parameters can be set in web.xml or using annotations.
2. **Service Phase**
   - Handles client requests.
   - service() method processes each request.
   - Differentiates between HTTP methods (GET, POST, etc.).
3. **Destruction Phase**
   - Servlet container removes the Servlet from memory.
   - destroy() method is called before the Servlet is destroyed.
   - Clean up resources and perform final operations.

Servlet API Overview

- **Understanding the Servlet API**
- The Servlet API provides classes and interfaces for developing Servlets.
- Part of the Java Enterprise Edition (Java EE) platform.
- Found in the javax.servlet and javax.servlet.http packages.

- **Key Classes and Interfaces**

✓ **Servlet Interface**

- Defines methods that all Servlets must implement.
- Includes init(), service(), and destroy() methods.

✓ **HttpServletRequest Interface**

- Represents an HTTP request from the client.
- Provides methods to access request parameters, headers, and attributes.

✓ **HttpServletResponse Interface**

- Represents an HTTP response sent back to the client.
- Provides methods to set response headers, status codes, and content.

Q & A

Thank You!

Deployment Of Servlets

SESSIONS/COOKIES

Servlets Overview

- **What are Servlets?**
  - Servlets are Java classes that extend the functionality of web servers to handle HTTP requests.
  - They are part of the Java EE (Enterprise Edition) platform, providing a powerful way to create dynamic web content.

- **Features of Servlets:**
  1. **Platform Independence:** Servlets run on any server that supports the Java EE specification, making them highly portable.
  2. **Dynamic Content Generation:** Servlets can generate dynamic content based on client requests, such as HTML, XML, JSON, etc.
  3. **Session Management:** Servlets support session management, allowing stateful interactions with clients.
  4. **Integration with Java EE Technologies:** Servlets seamlessly integrate with other Java EE technologies like JDBC for database access.

What are Sessions in Web Applications?

- Sessions in web applications refer to the mechanism of maintaining stateful interactions with clients.
- They enable the server to store and retrieve data associated with a specific client across multiple requests.

Importance of sessions

- Enable stateful interactions: Sessions allow applications to remember user preferences, shopping carts, login status, etc.
- Personalization: Customize user experience based on stored session data.
- Security: Sessions can be used for authentication and authorization purposes.

What is Session Management?

- Session management refers to the process of controlling and maintaining user sessions in web applications.
- It involves creating, tracking, and managing session data to enable stateful interactions with clients.

1. **Session Creation:** Initiating a session when a user accesses the application.
2. **Session Tracking:** Identifying and associating sessions with specific clients.
3. **Session Timeout:** Setting a duration after which inactive sessions are invalidated.
4. **Session Invalidation:** Ending a session either explicitly or due to timeout.

What are Cookies in Web Development?

- Cookies are small pieces of data stored on the client-side (in the user's browser) by websites.
- They are commonly used for session management, tracking user preferences, and personalizing user experiences.

- ✓ Use secure cookies (HTTPS) for transmitting sensitive data.
- ✓ Avoid storing sensitive information like passwords or authentication tokens in cookies.

What is State Management?

- State management in web applications refers to the process of preserving and managing data and application state across multiple user interactions.
- It involves storing and retrieving user-specific information, session data, and application state variables.

- **Types of State Management:**
  - **Client-Side State Management**
  - **Server-Side State Management**

Client-Side State Management

- **Client-Side State Management**
- Data stored on the client side (browser) using cookies, local storage, or session storage.
- Suitable for small amounts of data and maintaining user preferences.

- **Server-Side State Management:**
- Data stored on the server side, typically in session objects or databases.
- Used for managing user sessions, application state, and complex data interactions.

Client-Side State Management

1. **Cookies:**
   - Small pieces of data stored on the client's browser.
   - Used for storing session IDs, user preferences, and tracking information.
   - Limited in size and not suitable for storing large amounts of data or sensitive information.
2. **Local Storage:**
   - HTML5 API for storing key-value pairs locally on the client side.
   - Persistent storage that remains even after the browser is closed.
   - Ideal for storing application settings, user preferences, and cached data.
3. **Session Storage:**
   - Similar to local storage but cleared when the session ends (browser/tab is closed).
   - Useful for temporary data storage during a user's session.

Advantages of Client-Side State Management

- Reduced server load: Offloads data storage and management tasks to the client side.
- Faster access: Client-side data retrieval is often faster than server-side requests.
- Improved user experience: Enables personalized interactions and offline capabilities.

Server-Side State Management

1. **Session Objects:**
   - Server-side objects that store session data for each user session.
   - Managed by the server and accessible across multiple requests during a session.
   - Suitable for storing user authentication tokens, session-specific data, and shopping cart items.
2. **Databases:**
   - Persistent storage solutions for storing user data, application state, and session information.
   - Commonly used databases include MySQL, PostgreSQL, MongoDB, and Oracle.

Advantages of Server-Side State Management

- Enhanced security: Data stored on the server is more secure than client-side storage.
- Scalability: Server-side solutions can handle large amounts of data and concurrent users.
- Centralized management: Simplifies data access, retrieval, and manipulation on the server.

Q & A

Thank You!