**Day 1**

**11-10-2021**

**Servlet**

**Maven pom.xml file**

**LinkedList**

**Integration of github with Eclipse IDE**

**JSTL**

**Stack and Queue**

**Stream API**

**JDBC**

**Hibernate**

Java Database Connectivity

JDBC is a API (Application programming interface) which provide set of classes interface which help to connect the database through Java technologies.

Steps to connect the database

1. Import the packages java.sql.\* and javax.sql.\*;
2. Jdbc throw the checked exception. So we have to handle it using try-catch or throws.
3. Load the driver : it is pre-defined class which help to connect the database.

4 types of driver

1. JDBC Odbc bridge driver or type 1
2. JDBC net api driver or type 2
3. JDBC net protocol or type 3
4. JDBC pure or thin driver or type 4

From Java8 onwards type 1 driver removed.

Class.forName(“driverName”)

Build tool : Maven and Gradle

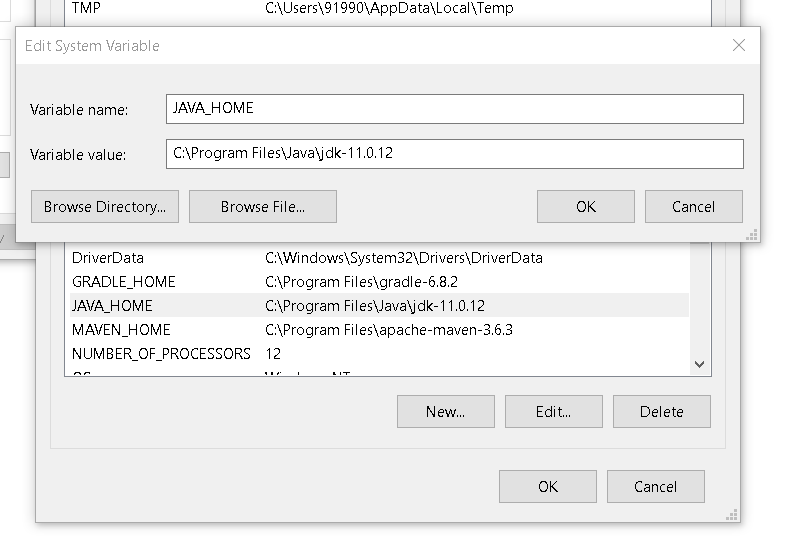
**Maven** : maven is a type of build tool. It is use to compile program, run program , creating jar, war or ear file, downloading the dependencies and creating the documentation.

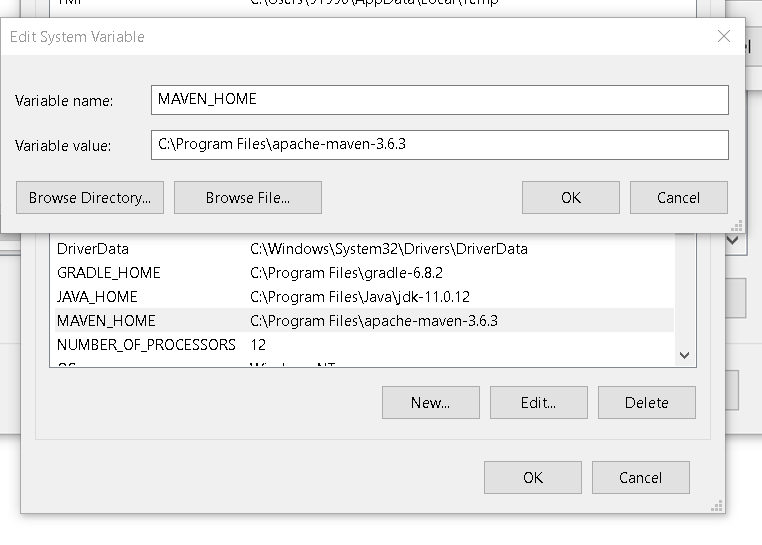
Maven use pom.xml file (project object model). This file is known as Maven configuration file.

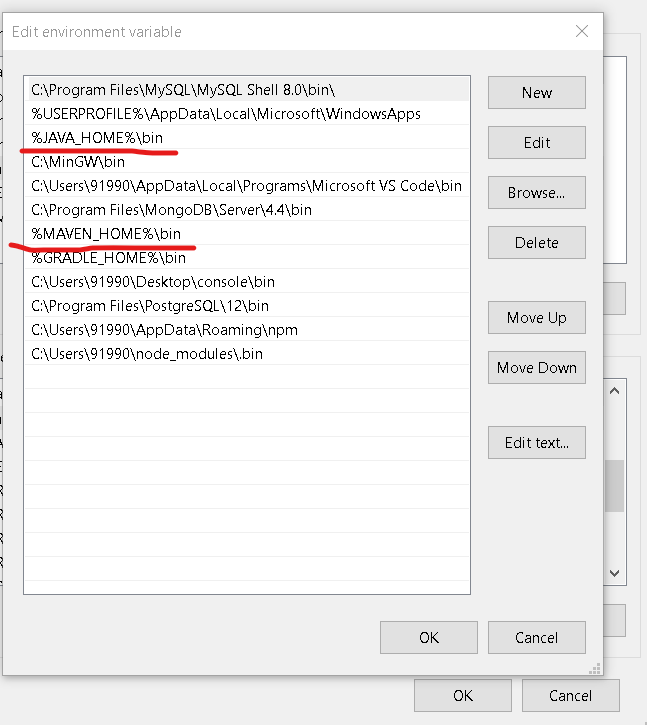
**Jar** : core java project .java and .class

**War** : web project : .html,.css,.js.,xml, (.java(Servlet)), .jsp,(JSTL) etc

**Ear** : enterprise : .html,.css,.js.,xml, (.java(Servlet)), .jsp and .java (EJB application).







mvn validate : This command is use to pom.xml file syntax.

mvn compile

mvn clean

mvn install

mvn test

MySQL

Student 🡪 Table : Database sid,name,age 🡪 columns

Student -🡪 Java Bean Class : Java Side sid,name,age->property

Main

Student ss = new Student();

ss.setId(100);

ss.setName(“Raj”);

ss.setAge(21);

Student ss1 = new Student(101,”Ramesh”,23);

Java Bean class is known as pure Encapsulation class.

Service class or service layer : This class is responsible to write business logic.

Pure business logic.

Business + database logic

Dao class or Dao layer : Data Access Object :

Pure database logic.

Database -🡪Student

Bean --🡪 Student

Service -🡪 Student business logic

Dao --🡪 Pure jdbc code to insert, delete, update and retrieve

Main class with main method which interact input device.

Dao and Service layer not responsible to interact with input device.

In record in database is equal to one object in Java side.

Do {

1: Add

2: display

3 : delete

4 : update

Switch() {

Case 1:

Case 2:

Case 3:

Case 4:

}

Do you want to continue

}while()

**Day 2**

**12-10-2021**

**resource layer : This layer is responsible to provide resource details like**

**database connectivity**

**file resources**

**security resource**

**etc**

**To make the resource layer we can use**

1. **Java class**
2. **Xml file**
3. **Properties**

**ORM : Object Relation Mapping**

**Limitation of JDBC**

1. **Using JDBC we can’t store as well as retrieve the object from a database directly. We have to convert object to query and vice-versa.**
2. **JDBC use SQL language. SQL is database dependent language.**
3. **JDBC throw checked exception. JDBC exception hierarchy is database dependent.**
4. **JDBC doesn’t support is a and has relationship.**

**Is a :inheritance**

**Has a:**

**class Employee {**

**Address add;**

**}**

**class Address {**

**}**

13-10-2021

ORM : Object Relation Mapping

Entity class

Java side (Object) relation

class Student { Student -🡪 Table sid,name,age SID,NAME,AGE—column

}

Student ss = new Student(); 100,Raj,21

ss.setSId(100);

ss.setName(“Raj”);

ss.setAge(21);

mapping

Student (className)—Student

sid-🡪SID pk data types

sname🡪 sname

age -🡪age

using xml file

using annotation

DemoTest

Main method

Hibernate API which help to do some operation on table.

Table create in database.

Entity class (like java bean )

Now have to create the mapping file

Or

Annotation

Class level

@Entity

@Id the column contains primary key.

Resource layer : DB connection

Java classes

Xml file : Hibernate configuration file

(hibernate.cfg.xml)

Properties

hibernate.cfg.xml

database details

driver name, url, username and password

dialects class : This class is responsible to convert java object to sql and vice-versa.

Inside this file we have to provide the details about entity class with mapping file or annotation.

Through JDBC we do any DML operation by default it auto commit.

But through ORM tools by default not auto commit.

TCML (Transactional control language).

Commit and rollback.

**SQL**

Structure Query language

Select \* from student : here student is table.

Retrieve all column from a student table.

**HQL**

Hibernate Query language

Select s from Student s : here Student is entity

class name case sensitive

retrieve all objects form a Student entity class.

view(html/jsp)-🡪servlet(controller)--🡪service -🡪 dao(orm)-🡪 resource (xml file)--🡪 mysql

**MVC Model View Controller**

View -🡪 HTML/JSP console class

Controller -🡪Servlet main class

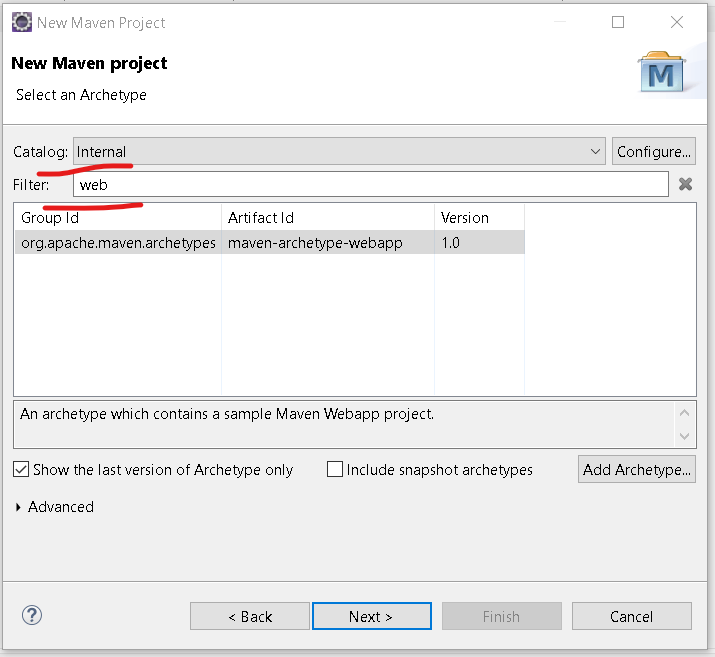
Model --🡪 Entity, service class, dao class, resource layer

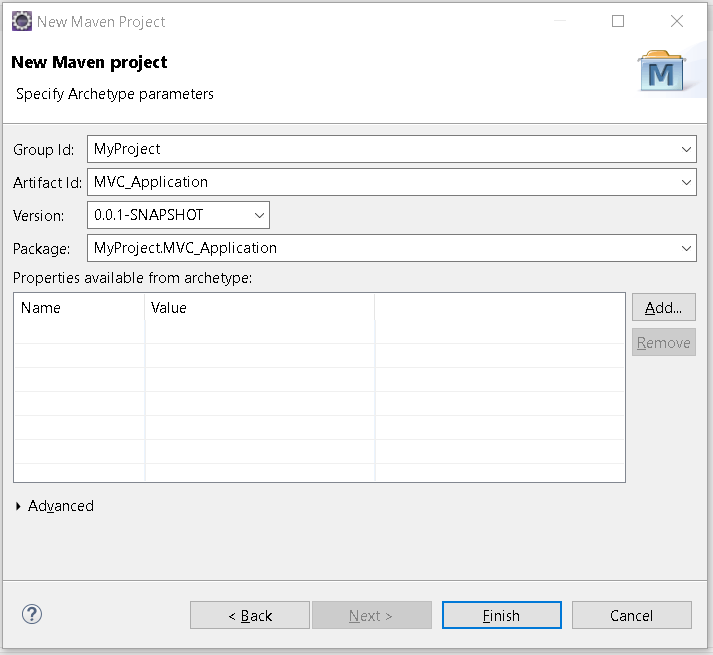
Day 3

14-10-2021

Creating maven web project

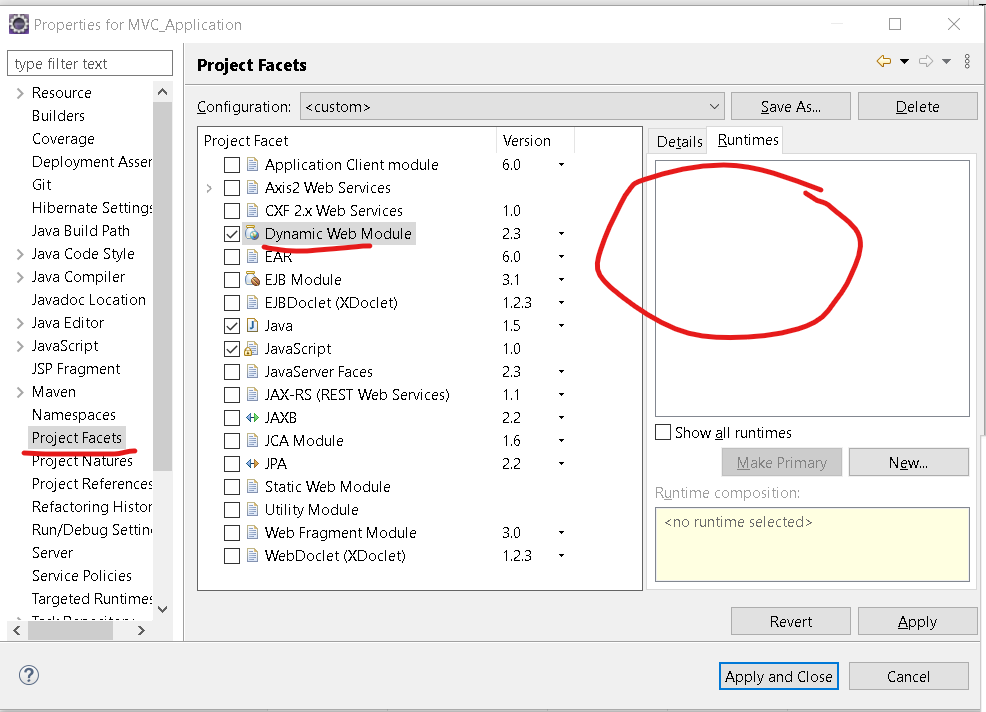
Don’t select create project option





After project create to add the tomcat

Right click on project and properties



In Maven web project we have to create two folder which help to add java and testing files.

From browser

<http://localhost:8080/projectName/urlPattern>

RequestDipatcher is a API provided by Servlet which help to navigate from one page to another page.

If we want to share the data between two pages ie may be servlet or JSP.

JEE (Java Enterprise Edition) provide concept called object scope

Page or servlet scope

Request scope

Session scope

Application scope

<%

Java coding

%>

<%=variableName%>

LoginService

Two method

signIn

signUp

LoginController

Two methods

doGet

Login Page login.jsp

doPost

Sign Up signUp.jsp