** MAVEN **

jar : To run any java project we must provide at least one jar to JDK(JVM). jar is a collection of .class files (or even it can contain any data like .java, .xml , .txt etc..)

All basic classes like String, Date, Thread, Exception, List etc.. are exist in rt.jar (runtime jar) given by Sun(Oracle). rt.jar will be set to project by JDK/JRE only. Programmer not required to set this jar.

Every jar is called as dependency. It means program depends on jars. Sometimes one jar may need another jar(child jar), this is called as dependency chain.

ex: Spring jars needs commons logging jar, hibernate jars needs slf4j,java-asst jar

Every jar directly or indirectly depends on rt.jar. Always parent jar version >= child jar version

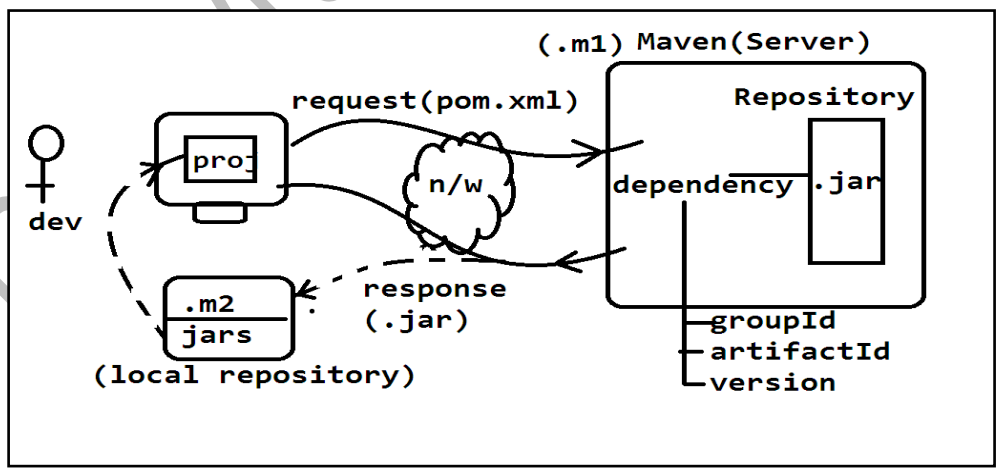
Dependency Management :

It is a concept of maintain jars of a project, maintain child jars and parent jars (dependency-chain), maintain proper version of all jars (dependency-version)

Library : It is collection of Jars.

Maven is a Tool (software) used to handle dependency-management as automated ie jars, chain jars with version management should be taken care by maven tool. It is developed by Apache. Maven is a Repository server which contains jars(dependencies) only in well-arranged format (provider based, jar name based, version based).

Every jar (depedency) contains 3 details mainly. Those are groupId (jar provider/company name) artifactId (jar name) & version (jar version). Maven maintains all jars as Library also called as Maven Library.



Maven project created in local system should make request. Request should be in XML format. That is pom.xml (pom= project object model). Maven server takes pom.xml request and returns jars as response. All jars will be copied to local repository first ie ".m2" folder.

Every jar(dependency)should contain,

groupId (providerName)
artifactId (jar name)
version (jar version).

Format looks as,
<dependency>
<groupId> ___ </groupId>
<artifactId>___ </artifactId>
<version> ____</version>
</dependency>

Maven Projects :

To create Maven Project we should choose one option given below.

1. Simple Project : Basic Application
2. ArchType Project : Like web-app, framework related etc...

For any Maven project 4 basic folder are provided. Those are,

- **src/main/java**
- **src/main/resource**
- **src/test/java**
- **src/test/resource**

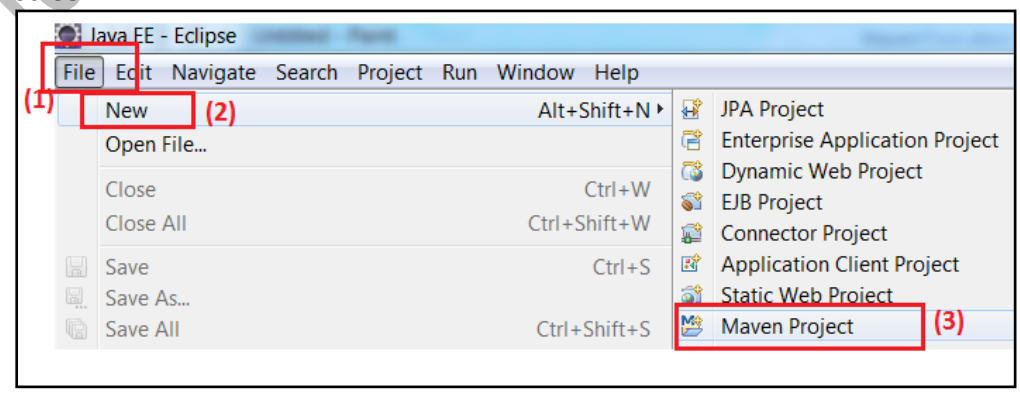
"java" folder is used to store only .java files, where "resources" are used to store non-java files like .properties, .xml, .txt, etc... "main" indicates files related to project developement. "test" indicates files related to UnitTesting.

Creating Simple Maven Project :

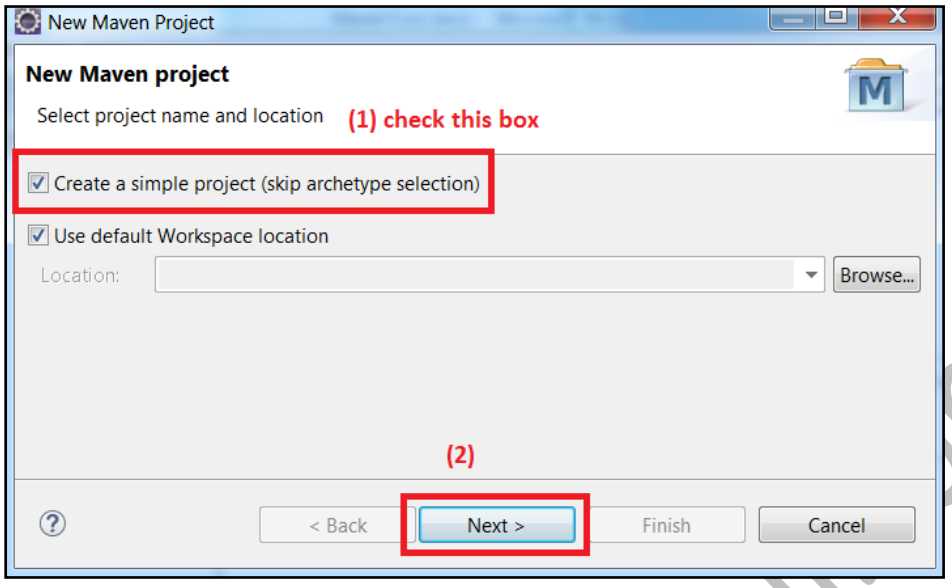
This type of projects are used in java to write JDBC,Hibernate,Spring core or any stand alone applications only. Steps to create Simple Maven Project:-

1. click on "File" menu
2. click on "New" option
3. Choose "Maven Project" option. If not exist goto "other", search using maven project word.
4. Screen looks as below. Here select checkbox "Create Simple Project".
5. on click next button, maven project needs groupId,artifactId,version details of our project.

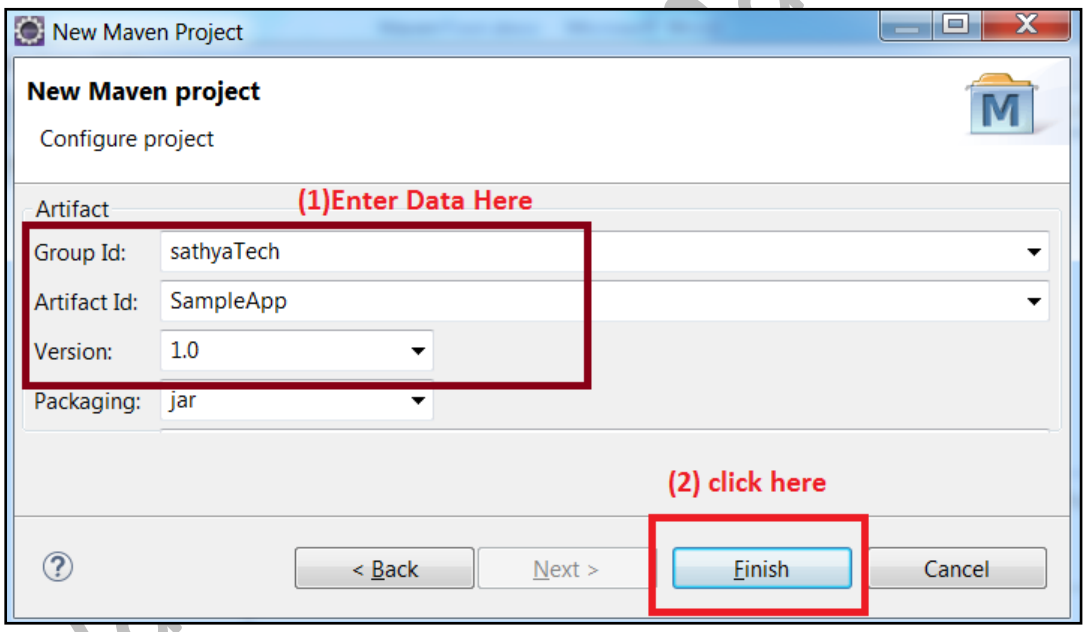
Screen : 1



Screen:2



Screen:3



At last our project also be converted/exported as .jar (java archive), .war(web archive) or .ear (enterprise archive). Here version can contain numbers, characters and symbols. (Screen looks as below)

Setup JDK/JRE in Eclipse :

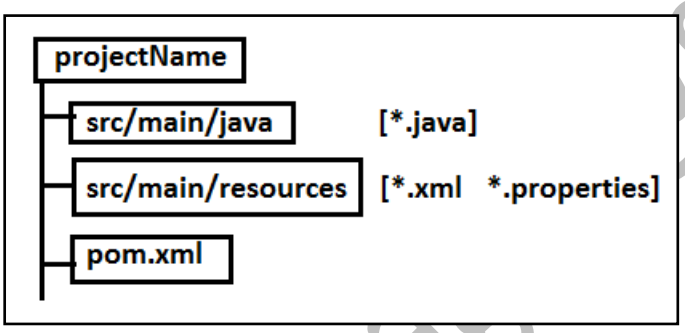
A. Window menu
B. preferences option
C. Search with "installed JRE"
D. choose Installed JRE under Java
E. click on "Add" Button to add another
F. Select "Standard VM "
G. next

H. Click on "Directory/Browse"
I. select location (ex: C:/Program Files/Java/jdk1.8)
J. finish/ok
K. finish/ok

Now Modify JRE Version to Maven

a) Right click on maven project
b) Build path option
c) Click on configure Build path..
d) Choose Library Tab
e) Select/click on "JRE System Library"
f) Click on "Edit" Button
g) Choose any one JRE
h) Finish
i) ok

Basic Folder structure for Simple Maven Project :



✓ Open pom.xml file in XML View mode. In this root tag is <project>
✓ Under this tag, add one tag
<dependencies> </dependencies>

Then file looks like

```
<project ....>
 <modelVersion>4.0.0</modelVersion>
 <groupId> ------- </groupId>
 <artifactId> ---- </artifactId>
 <version> ------- </version>
 <dependencies>



 </dependencies>
</project>
```

Here, we must specify jar details under <dependencies> tag example : for spring core programming

```
<dependency>
 <groupId>org.springframework</groupId>
 <artifactId>spring-context</artifactId>
 <version>4.3.2.RELEASE</version>
</dependency>
```

For mysql-connector jar :

```
<dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <version>5.1.26</version>
</dependency>
```

*** User Library ***

If any Jar not exist in Maven or unable get from Maven Server (ex: created by One company or by programmer) then we can add jar to project using this concept. User Library has two steps,

1. Create Library in Workspace

2. Add Library to project.

1. Create Library in Workspace

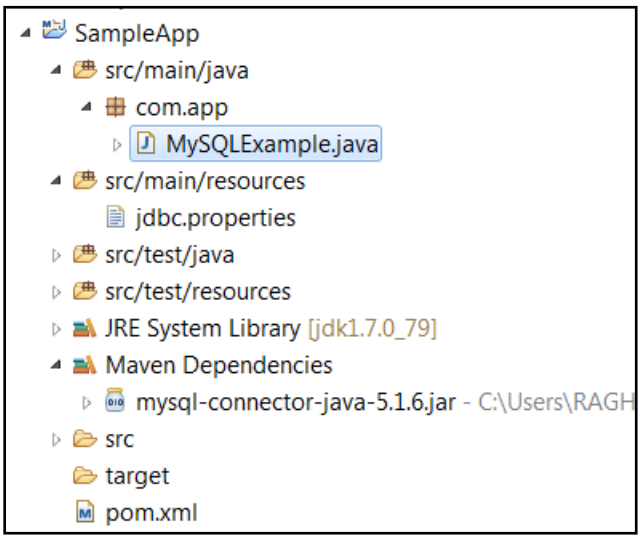
i. Window menu
ii. Preference option
iii. Search with "User Library" word
iv. Select User Library under Java
v. Click on New button
vi. Enter Library Name (ex: MyLibs)
vii. Finish/ok
viii. Click on Library Name
ix. Click on Add External Jars Button
x. Select Jars required from folders
xi. Finish
xii. ok

2. Add Library to project:

I. Right click on project
II. Choose "Build path"
III. Configure Build path ...
IV. Click on Library Tab.
V. Click on "Add Library" Button
VI. Select "User Library" option
VII. Next button
VIII. Choose "MyLibs"
IX. Click on Finish and ok.

EXAMPLE MAVEN APPLICATIONS

1. JDBC USING MYSQL TYPE-4 DRIVER CLASS WITH PROPERTIES FILE



pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <groupId>sathyaTech</groupId>
      <artifactId>SampleApp</artifactId>
      <version>1.0</version>
      <dependencies>
            <dependency>
                  <groupId>mysql</groupId>
                  <artifactId>mysql-connector-java</artifactId>
                  <version>5.1.6</version>
            </dependency>
      </dependencies>
</project>
```

Step#1 Right click on src/main/resources folder , choose here new->other option. Search with "file" and choose same. Enter File name as "jdbc.properties".

```
# JDBC MySQL Connection properties

jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/test
jdbc.user=root
jdbc.password=root
```

Step#2 Create one java Class "MySQLExample.java". Here defined 3 static methods to load properties file, to get DB Connection and to select data from employee table.

```
package com.app;

import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

/**
 * @author RAGHU
 * @version JDK 1.7
 */

public class MySQLExample {

	/**
	 * 1. Load Connection Details from Properties file
	 */
	public static Properties loadProperties(){
		Properties props=null;
		try {
			ClassLoader loader = Thread.currentThread().
					getContextClassLoader();
			InputStream resourceStream = loader.
				getResourceAsStream("jdbc.properties");
			props=new Properties();
			props.load(resourceStream);

		} catch (Exception e) {
			e.printStackTrace();
		}
		return props;
	}

	/**
	 * 2. Get Connection
	 *  @throws Exception
	 */
	public static Connection getConnection() throws Exception{

			Properties props=loadProperties();
			Class.forName(props.getProperty("jdbc.driver"));
			Connection con = DriverManager.getConnection(
					props.getProperty("jdbc.url"),
					props.getProperty("jdbc.user"),
					props.getProperty("jdbc.password"));
			return con;
	}
	/**
	 * 3. Select Data from employee table
	 * @throws Exception
	 */
	public static void selectEmployee() throws Exception{
```

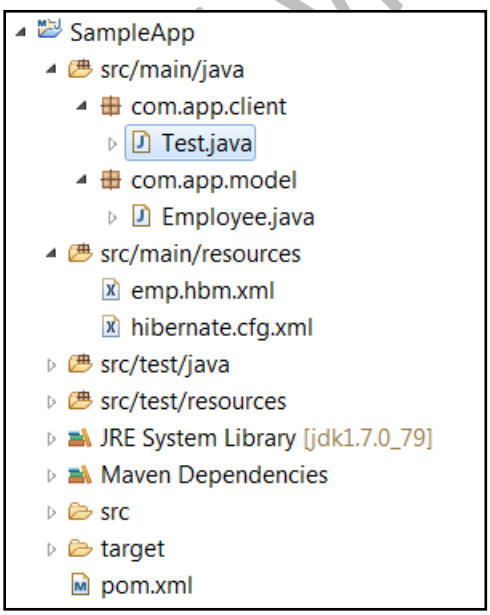
```
			Statement st = getConnection().createStatement();
			ResultSet rs = st.executeQuery("select * from employee");
			while (rs.next()) {
					System.out.print(rs.getInt(1)+",");
					System.out.print(rs.getString(2)+",");
					System.out.println(rs.getDouble(3));
				}

	}
	public static void main(String[] args) {
			try {
				selectEmployee();
			} catch (Exception e) {
				e.printStackTrace();
			}
	}
}
```

Create Table & insert Records:

```
employee  CREATE TABLE employee (
           empId int(11) NOT NULL,
           empName varchar(255) DEFAULT NULL,
           empSal double NOT NULL,

         )  ;
insert into employee values(10,'AA',55.23);
insert into employee values(11,'BB',66.24);
```

2. HIBERNATE EXAMPLE APPLICATION USING MYSQL DATABASE.

Folder Structure:-



pom.xml file

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <groupId>sathyaTech</groupId>
      <artifactId>SampleApp</artifactId>
      <version>1.0</version>
      <dependencies>
        <dependency>
           <groupId>org.hibernate</groupId>
           <artifactId>hibernate-core</artifactId>
           <version>4.3.5.Final</version>
        </dependency>
        <dependency>
           <groupId>mysql</groupId>
           <artifactId>mysql-connector-java</artifactId>
           <version>5.1.6</version>
        </dependency>
       </dependencies>
</project>
```

Model class : Employee.java

```
package com.app.model;

public class Employee {

      private int empId;
      private String empName;
      private double empSal;

      //alt+shift+O (De-select All->OK)
      public Employee() {
      }
      //alt+Shift+S , R (select all) ->OK
      public int getEmpId() {
            return empId;
      }
      public void setEmpId(int empId) {
            this.empId = empId;
      }
      public String getEmpName() {
            return empName;
      }
      public void setEmpName(String empName) {
            this.empName = empName;
      }
      public double getEmpSal() {
            return empSal;
      }
```

```
	public void setEmpSal(double empSal) {
		this.empSal = empSal;
	}
	//alt+shift+s,s ->ok
	@Override
	public String toString() {
		return "Employee [empId=" + empId + ", empName=" + empName +
", empSal=" + empSal + "]";
	}



}
```

Mapping File : emp.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
	<class name="com.app.model.Employee" table="EMPLOYEE">
		<id name="empId" column="eid"/>
		<property name="empName" column="ename"/>
		<property name="empSal" column="esal"/>
	</class>
</hibernate-mapping>
```

configuration file hibernate.cfg.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
	<session-factory>
		<property name="hibernate.connection.driver_class">
			com.mysql.jdbc.Driver </property>
<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/test
</property>
 <property name="hibernate.connection.user">root</property>
 <property name="hibernate.connection.password">root</property>

 <property name="hibernate.show_sql">true </property>
<property name="hibernate.dialet">org.hibernate.dialect.MySQLDialect
</property>
<property name="hibernate.hbm2ddl.auto">create</property>
<mapping resource="emp.hbm.xml" />
 </session-factory>
</hibernate-configuration>
```

Test class :

```
package com.app.client;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

import com.app.model.Employee;

public class Test {

	public static void main(String[] args) {
		Configuration cfg = new Configuration();
		cfg.configure(); //loads hibernate.cfg.xml file

		/*SessionFactory sf = cfg.buildSessionFactory();*/
		SessionFactory sf = cfg.buildSessionFactory(
				new StandardServiceRegistryBuilder().
				applySettings(cfg.getProperties()).build());

		Session ses = sf.openSession();
		Transaction tx = ses.beginTransaction();

		Employee emp=new Employee();
		emp.setEmpId(100);
		emp.setEmpName("ABC");
		emp.setEmpSal(12.36);

		ses.save(emp);

		tx.commit();
		ses.close();

		System.out.println("Record inserted..");
	}
}
```

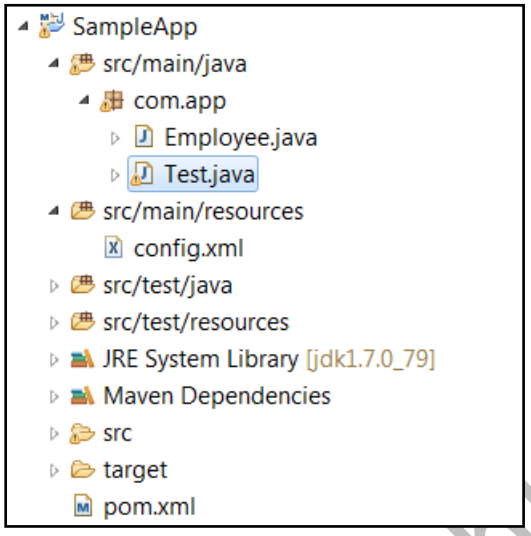

3. SPRING CORE EXAMPLE :

pom.xml file

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>sathyaTech</groupId>
```

```
	<artifactId>SampleApp</artifactId>
	<version>1.0</version>
	<dependencies>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-context</artifactId>
			<version>4.3.4.RELEASE</version>
		</dependency>
	</dependencies>
</project>
```

Folder Structure :



Spring Bean:

```
package com.app;

public class Employee {

	private int empId;
	private String empName;
	private double empSal;

	public int getEmpId() {
		return empId;
	}
	public void setEmpId(int empId) {
		this.empId = empId;
	}
	public String getEmpName() {
		return empName;
	}
	public void setEmpName(String empName) {
		this.empName = empName;
	}
	public double getEmpSal() {
		return empSal;
```

```
	}
	public void setEmpSal(double empSal) {
		this.empSal = empSal;
	}
	@Override
	public String toString() {
		return "Employee [empId=" + empId + ", empName=" + empName +
", empSal=" + empSal + "]";
	}

}
```

Spring Configuration File (XML File) : config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
   ">

	<bean class="com.app.Employee" name="empObj">
		<property name="empId" value="100"/>
		<property name="empName" value="ABC"/>
		<property name="empSal" value="2.36"/>

	</bean>

</beans>
```

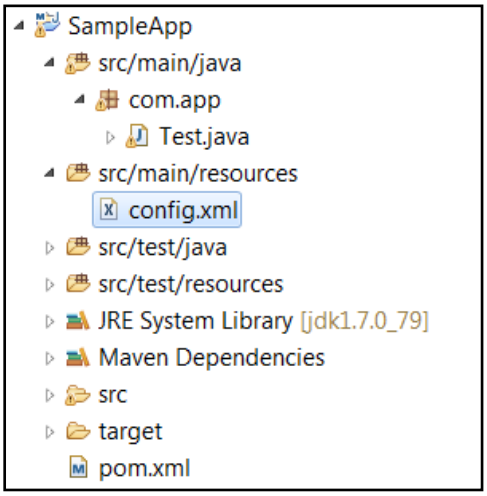
Test class:

```
package com.app;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
 * @author RAGHU
 * @version JDK 1.7
 */

public class Test {

	public static void main(String[] args) {
		ApplicationContext context=new
ClassPathXmlApplicationContext("config.xml");
		Object ob=context.getBean("empObj");
		if(ob instanceof Employee){
			Employee emp=(Employee) ob;
			System.out.println(emp);
		}else{
			System.out.println("object is not employee type");
```

```
		}

	}
}
```

4. SPRING JDBC EXAMPLE :

Folder Structure:



pom.xml file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>sathyaTech</groupId>
	<artifactId>SampleApp</artifactId>
	<version>1.0</version>
	<dependencies>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-context</artifactId>
			<version>4.3.4.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-jdbc</artifactId>
			<version>4.3.4.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<version>5.1.19</version>
		</dependency>
```

```
	</dependencies>
</project>
```

Spring Configuration File (XML File): config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:p="http://www.springframework.org/schema/p"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
   ">

	<bean
class="org.springframework.jdbc.datasource.DriverManagerDataSource"
		name="dataSourceObj"
		p:driverClassName="com.mysql.jdbc.Driver"
		p:url="jdbc:mysql://localhost:3306/test"
		p:username="root"
		p:password="root"
	/>

	<bean class="org.springframework.jdbc.core.JdbcTemplate"
name="jtObj"
		p:dataSource-ref="dataSourceObj"
	/>

</beans>
```

Test class:

```
package com.app;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.jdbc.core.JdbcTemplate;

/**
 * @author RAGHU
 * @version JDK 1.7
 */

public class Test {

	public static void main(String[] args) {
		ApplicationContext context=new
ClassPathXmlApplicationContext("config.xml");
		JdbcTemplate jt=(JdbcTemplate) context.getBean("jtObj");
		String sql="insert into employee values(?,?,?)";
		int count=jt.update(sql, 20,"ABC",36.36);
		System.out.println(count);
```

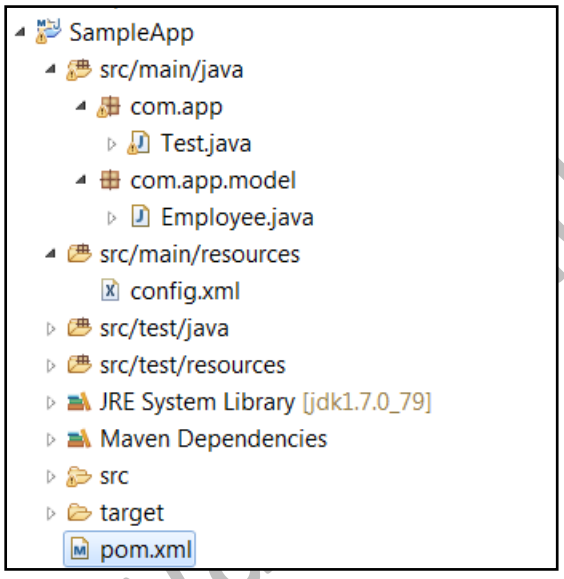
```
	}
}
```

Create Table:

```
CREATE TABLE employee (
		empId int(11) NOT NULL,
		empName varchar(255) DEFAULT NULL,
		empSal double NOT NULL,

)  ;
```

5. SPRING ORM EXAMPLE USING MYSQL DATABASE:

Folder Structure :



pom.xml file

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>sathyaTech</groupId>
	<artifactId>SampleApp</artifactId>
	<version>1.0</version>
	<dependencies>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-context</artifactId>
			<version>3.2.6.RELEASE</version>
```

```
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-orm</artifactId>
			<version>3.2.6.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-core</artifactId>
			<version>3.6.4.Final</version>
		</dependency>
		<dependency>
			<groupId>javassist</groupId>
			<artifactId>javassist</artifactId>
			<version>3.12.1.GA</version>
		</dependency>
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<version>5.1.19</version>
		</dependency>
	</dependencies>
</project>
```

Model class : Employee.java

```
package com.app.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="employee")
public class Employee {
	@Id
	@Column(name="eid")
	private int empId;
	@Column(name="ename")
	private String empName;
	@Column(name="esal")
	private double empSal;

	public int getEmpId() {
		return empId;
	}
	public void setEmpId(int empId) {
		this.empId = empId;
	}
	public String getEmpName() {
		return empName;
	}
```

```
	public void setEmpName(String empName) {
		this.empName = empName;
	}
	public double getEmpSal() {
		return empSal;
	}
	public void setEmpSal(double empSal) {
		this.empSal = empSal;
	}
	@Override
	public String toString() {
		return "Employee [empId=" + empId + ", empName=" + empName +
", empSal=" + empSal + "]";
	}



}
```

Spring Configuration File (XML File) config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:p="http://www.springframework.org/schema/p"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
   ">

	<bean
class="org.springframework.jdbc.datasource.DriverManagerDataSource"
		name="dataSourceObj"
		p:driverClassName="com.mysql.jdbc.Driver"
		p:url="jdbc:mysql://localhost:3306/test"
		p:username="root"
		p:password="root"
	/>

	<bean
	class="org.springframework.orm.hibernate3.annotation.AnnotationSessi
onFactoryBean"		name="sfObj">
		<property name="dataSource" ref="dataSourceObj"/>
		<property name="hibernateProperties">
			<props>
				<prop
key="hibernate.dialect">org.hibernate.dialect.OracleDialect</prop>
				<prop key="hibernate.show_sql">true</prop>
				<prop key="hibernate.hbm2ddl.auto">create</prop>
			</props>
		</property>
		<property name="annotatedClasses">
			<list>
				<value>com.app.model.Employee</value>
```

```
                </list>
            </property>
      </bean>

      <bean class="org.springframework.orm.hibernate3.HibernateTemplate"
            name="htObj" p:sessionFactory-ref="sfObj"/>

</beans>
```

Test class:

```
package com.app;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.orm.hibernate3.HibernateTemplate;

import com.app.model.Employee;

/**
 * @author RAGHU
 * @version JDK 1.7
 */

public class Test {

      public static void main(String[] args) {
            ApplicationContext context=new
ClassPathXmlApplicationContext("config.xml");
            HibernateTemplate ht=(HibernateTemplate)
context.getBean("htObj");

            Employee emp=new Employee();
            emp.setEmpId(123);
            emp.setEmpName("ABC");
            emp.setEmpSal(12.36);
            ht.save(emp);
            System.out.println("Saved successfully..");


      }
}
```