Raghu Sir (Naresh i Technologies,Ameerpet-Hyderabad)

Gradle

Gradle is a build system (open source) which is used to automate building, testing, deployment etc. "build. **gradle**" are scripts where one can automate the tasks.

- **Fast** — Gradle completes tasks quickly by reusing outputs from previous executions, processing only inputs that changed, and executing tasks in parallel.
- **Highly customizable** — Gradle is modeled in a way that is customizable and extensible in the most fundamental ways.
- **Powerful** — Gradle is the official build tool for Android, and comes with support for many popular languages and technologies.

Widely used at: Java, Android, C++, Kotlin, Groovey Scala and Java Script

Maven	Gradle
It uses an XML file for declaring the project, its dependencies, the build order, and its required plug-in.	It is a build automation system that uses a Groovy-based DSL (domain-specific language)
Goal is related to the project phase.	Goal is to add functionality to the project.
It does not use the build cache; thus, its build time is slower than Gradle.	Runs the tasks that have been changed. Therefore it gives a faster performance.
Customization is a bit complicated.	Gradle is highly customizable
The compilation is mandatory in Maven.	Gradle avoids the compilation of Java.

JDBC Example

1. build.gradle

```
plugins {
      id 'java'
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

repositories {
   mavenCentral()
```

```
}

dependencies {
  //implementation group: 'mysql', name: 'mysql-connector-java', version:
'8.0.25'
  implementation 'mysql:mysql-connector-java:8.0.25'

}
```

Test.java

```
package in.nit.raghu;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class Test {

	private static String driver="com.mysql.cj.jdbc.Driver";
	private static String url="jdbc:mysql://localhost:3306/test";
	private static String username="root";
	private static String password="root";
	private static String sql="insert into student values(?,?,?)";

	public static void main(String[] args) throws Exception {
		Class.forName(driver);
		Connection con = DriverManager.getConnection(url,username,password);
		PreparedStatement pstmt=con.prepareStatement(sql);
		pstmt.setInt(1,10);
		pstmt.setString(2,"AJ");
		pstmt.setDouble(3,3.3);
		int count=pstmt.executeUpdate();
		System.out.println("Inserted:"+count);
		con.close();
	}
}
```

Hibernate Application

1. **build.gradle**

```
plugins {
    id 'java'
}

repositories {
     mavenCentral()
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

dependencies {
     implementation group: 'org.hibernate', name: 'hibernate-core', version:
'5.4.32.Final'
     compileOnly group: 'org.projectlombok', name: 'lombok', version: '1.18.20'
     annotationProcessor 'org.projectlombok:lombok:1.18.20'
     runtimeOnly group: 'mysql', name: 'mysql-connector-java', version:
'8.0.25'
}

/*jar {
    manifest {
        attributes 'Main-Class': 'in.nareshit.raghu.Test'
    }

}*/

task fatJar(type: Jar) {
    manifest {
       attributes 'Main-Class': 'in.nareshit.raghu.Test'
    }
    baseName = 'all-in-one-jar'
    from { configurations.runtimeClasspath.collect { it.isDirectory() ? it :
zipTree(it) } }
    with jar
}
```

2. **Entity class**

```
package in.nareshit.raghu;
```

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

import lombok.Data;

@Data
@Entity
@Table(name="prodtab")
public class Product {
	@Id
	@Column(name="pid")
	private Integer prodId;

	@Column(name="pcode")
	private String prodCode;

	@Column(name="pcost")
	private double prodCost;
}
```

3. hibernate.cfg.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
	<session-factory>
		<property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
		<property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/nit</property>
		<property name="hibernate.connection.username">root</property>
		<property name="hibernate.connection.password">root</property>
		<property name="hiberante.dialect">
org.hibernate.dialect.MySQL8Dialect</property>
		<property name="hibernate.show_sql">true</property>
		<property name="hibernate.format_sql">true</property>
		<property name="hibernate.hbm2ddl.auto">create</property>
		<mapping class="in.nareshit.raghu.Product" />
	</session-factory>
</hibernate-configuration>
```

4. Test class

```
package in.nareshit.raghu;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

public class Test {

	public static void main(String[] args) {
		Configuration cfg=new Configuration();
		cfg.configure();
		SessionFactory sf=cfg.buildSessionFactory();
		Session ses=sf.openSession();
		Transaction tx=ses.beginTransaction();

		Product p = new Product();
		p.setProdId(100);
		p.setProdCode("PEN");
		p.setProdCost(200.0);

		ses.saveOrUpdate(p);
		tx.commit();
		ses.close();
	}
}
```

Servlets Application

1. build.gradle

```
plugins {
    id 'java'
    id 'war'
    id "org.gretty" version "3.0.5"
}

repositories {
```

```
      mavenCentral()
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

dependencies {
      compileOnly group: 'javax.servlet', name: 'javax.servlet-api', version:
'3.1.0'
}

war {
    archiveName = 'sample.war'
}

gretty {
    httpPort  = 8080
    contextPath = '/'
    servletContainer = 'jetty9'
}
```

2. Welcome Servlet

```
package in.nareshit.raghu;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/welcome")
public class WelcomeServlet extends HttpServlet {

      private static final long serialVersionUID = 1L;

      protected void doGet(
                  HttpServletRequest req,
                  HttpServletResponse resp)
                  throws ServletException, IOException
      {
```

```
            PrintWriter out = resp.getWriter();
            out.print("<h3>WELCOME TO GRADLE APP</h3>");
      }
}
```

Spring Core Example

1. **build.gradle**

```
plugins {
   id 'java'
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

repositories {
   //jcenter()
   mavenCentral()
}

dependencies {
   //spring context
   implementation group: 'org.springframework', name: 'spring-context', version: '5.3.6'
   // project lombok
   compileOnly group: 'org.projectlombok', name: 'lombok', version: '1.18.20'
}
```

2. **Spring Configuration XML File**

```
<?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.xsd">

   <bean class="in.nareshit.raghu.Employee" id="empObj">
```

```
        <property name="empId" value="101"/>
        <property name="empName" value="AA"/>
        <property name="empSal" value="200.0"/>
    </bean>

</beans>
```

3. Spring Bean

```
package in.nareshit.raghu;

import lombok.Data;

@Data
public class Employee {

        private Integer empId;
        private String empName;
        private Double empSal;
}
```

4. Test class

```
package in.nareshit.raghu;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

        public static void main(String[] args) {
                ClassPathXmlApplicationContext ac = new ClassPathXmlApplicationContext("config.xml");
                Employee e = ac.getBean("empObj",Employee.class);
                System.out.println(e);
                ac.close();
        }
}
```