JAXB Example

- **Java Architecture for XML Binding** (**JAXB**) provides API for converting Object to XML and XML to Object easily. JAXB was developed as a separate project but it was used widely and finally became part of JDK in Java 6.
  JAXB Marshalling: Converting a Java Object to XML.
  JAXB Unmarhsalling: Converting XML to Java Object.
  Using JAXB is very easy and it uses java annotations. We need to annotate Java Object to provide instructions for XML creation and then we have to create Marshaller to convert Object to XML.
  Unmarshaller is used to convert XML to java Object.
  JAXBContext is the entry point for JAXB and provides methods to get marshaller and unmarshaller object.
- Some basic and useful JAXB annotations are:
  1. **@XmlRootElement**: This is a must have annotation for the Object to be used in JAXB. It defines the root element for the XML content.
  2. **@XmlType**: It maps the class to the XML schema type. We can use it for ordering the elements in the XML.
  3. **@XmlTransient**: This will make sure that the Object property is not written to the XML.
  4. **@XmlAttribute**: This will create the Object property as attribute.
  5. **@XmlElement(name = "abc"):** This will create the element with name "abc".

- **JAXB Example**
  It is a simple JAXB example program to demonstrate the use of JAXB marshalling and unmarshalling.

```
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
import javax.xml.bind.annotation.XmlType;

@XmlRootElement(name = "Emp")
@XmlType(propOrder = {"name", "age", "role", "gender"})
public class Employee {
```

```
    private int id;

    private String gender;

    private int age;
    private String name;
    private String role;

    private String password;
    @XmlTransient
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    @XmlAttribute
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }
    @XmlElement(name = "gen")
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public String getRole() {
        return role;
    }
    public void setRole(String role) {
        this.role = role;
    }
    @Override
    public String toString() {
        return "ID = " + id + " NAME=" + name + "
AGE=" + age + " GENDER=" + gender + " ROLE=" +
                role + " PASSWORD=" + password;
    }

}
```

Employee is a normal java bean with private fields and their getters and setters. Notice the use of JAXB annotations to define root element, element name, elements order and transient property.

Here is a test JAXB example program showing JAXB Marshalling and JAXB Unmarshalling.

```
import java.io.File;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
public class JAXBExample {

    private static final String FILE_NAME = "jaxb-
emp.xml";
    public static void main(String[] args) {
        Employee emp = new Employee();
        emp.setId(1);
        emp.setAge(25);
        emp.setName("Pankaj");
        emp.setGender("Male");
        emp.setRole("Developer");
        emp.setPassword("sensitive");
        jaxbObjectToXML(emp);

        Employee empFromFile = jaxbXMLToObject();
        System.out.println(empFromFile.toString());
    }
```

```
    private static Employee jaxbXMLToObject() {
        try {
            JAXBContext context =
JAXBContext.newInstance(Employee.class);
            Unmarshaller un =
context.createUnmarshaller();
            Employee emp = (Employee) un.unmarshal(new
File(FILE_NAME));
            return emp;
        } catch (JAXBException e) {
            e.printStackTrace();
        }
        return null;
    }
    private static void jaxbObjectToXML(Employee emp) {

        try {
            JAXBContext context =
JAXBContext.newInstance(Employee.class);
            Marshaller m = context.createMarshaller();
            //for pretty-print XML in JAXB
m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
            // Write to System.out for debugging
            // m.marshal(emp, System.out);
            // Write to File
            m.marshal(emp, new File(FILE_NAME));
        } catch (JAXBException e) {
            e.printStackTrace();
        }
    }

}
```

Above program creates following jaxb-emp.xml file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Emp id="1">
    <name>Pankaj</name>
    <age>25</age>
    <role>Developer</role>
    <gen>Male</gen>
</Emp>
```

Check that XML file don't have password field and we get following output when same XML file is unmarshalled to Object.

```
ID = 1 NAME=Pankaj AGE=25 GENDER=Male ROLE=Developer
PASSWORD=null
```