XML & JAX-B
##########

-> XML stands for Extensible Markup Language

-> XML is free & open source

-> XML is intereoperable (Language independent & Platform indenpendent)

-> XML we can use to transfer data from one application to another application

-> XML introduced by w3c org

-> The initial version of xml is 1.0 and the current version of xml is also 1.0

-> XML will represent data in the form of elements

-> An element is the combination of start tag and end tag

Ex: <name> Ashok IT </name>

-> We will have 2 types of elements in the XML

1) Simple Element

2) Compound Element

-> The element which contains data directley is called as Simple Element

```
<name> Ashok IT </name>

<type> Educational </type>
```

-> The element which contains child element(s) is called as Compound Element

```
<person>
        <id> 101 </id>
        <name> raju </name>
</person>
```

Note: here <person> is a compound element and <id> <name> are simple elements

-> We can have attributes also for the element

```
<student  branch="CSE">
        <id> 101 </id>
        <name> Mahesh </name>
</student>
```

Note: XML should have only one root element. Inside the root element we can have multiple child elements

```
<persons>
		<person>
			<id> 101 </id>
			<name> raju </name>
		</person>

		<person>
			<id> 101 </id>
			<name> raju </name>
		</person>
</persons>
```

```
############
JAX-B
#############
```

-> JAX-B stands for Java Architecture For XML Binding

-> JAX-B is used to convert Java object to xml and xml to java object

-> JAX-B is free and open source

-> JAX-B given by sun microsystem

-> JAX-B is part of JDK upto 1.8v

-> If you are using JDK 1.8+ version of java then you need to add JAX-B dependency in pom.xml file

-> The process of converting Java Object into xml is called as "Marshalling"

-> The process of converting XML data to Java Object is called as "Un-Marshalling"

-> To perform Marshalling and Un-Marshalling We need to design Binding Classes.

-> The java class which represents the structure of XML is called as Binding class.

-> JAX-B provided annotations to represent java class as Binding Class.

Note: Binding Class creation is one time operation.

Note: Earlier people used to create Binding Classes using XSD. XSD represents structure of xml.

```
#####################
Marshalling Example
#####################


@Data
@XmlRootElement
```

```
public class Person {

	private Integer id;
	private String name;
	private Integer age;
	private Long phno;
	private Address adress;
}
---------------------------------
@Data
public class Address {

	private String city;
	private String state;
	private String country;
}
---------------------------------------
public class ConverJavaToXml {

	public static void main(String[] args) throws Exception {

		Address addr = new Address();
		addr.setCity("Hyd");
		addr.setState("TG");
		addr.setCountry("India");

		Person person = new Person();
		person.setId(101);
		person.setName("John");
		person.setAge(25);
		person.setPhno(125757571);
		person.setAdress(addr);

		JAXBContext instance = JAXBContext.newInstance(Person.class);

		Marshaller marshaller = instance.createMarshaller();

		marshaller.marshal(person, new File("Person.xml"));

		System.out.println("Marshalling Completed....");
	}
}
-------------------------------------------------------------------------------
```

@XmlAccessorType(XmlAccessType.FIELD) : Controls marshalling and un-marshalling using fields of entity class

@XmlAccessorOrder : Follow order of variables in the class to marshall and un-marshall

@XmlElement(name = "PhoneNum") : It is used to change the name of element

@XmlAttribute : It represents variable as attribute in xml

@XmlTransient : To skip a variable in marshalling

Note: By default every variable will be considered as Element and variable name will be considered as element name.

```
@Data
@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
@XmlAccessorOrder
public class Person {

	private Integer id;
	private String name;

	@XmlTransient
	private Integer age;

	@XmlElement(name = "PhoneNum")
	private Long phno;

	@XmlAttribute
	private String type;

	private Address adress;
}
```

```
public class ConvertXmlToJava {

	public static void main(String[] args) throws Exception {

		File xmlfile = new File("Person.xml");

		JAXBContext context = JAXBContext.newInstance(Person.class);

		Unmarshaller unmarshaller = context.createUnmarshaller();

		Object object = unmarshaller.unmarshal(xmlfile);

		Person person = (Person) object;

		System.out.println(person);
	}
}
```
