XSTREAM - ANNOTATIONS

XStream supports annotations similarly like automatic configuration instead of coding. In the previous chapter, we've seen the following configurations in code.

```
xstream.alias("student", Student.class);
xstream.alias("note", Note.class);

xstream.useAttributeFor(Student.class, "studentName");
xstream.aliasField("name", Student.class, "studentName");
xstream.addImplicitCollection(Student.class, "notes");
```

The following code snippet illustrates the use of annotations to do the same work in a much easier way.

```
@XStreamAlias("student")   //define class level alias
class Student {

   @XStreamAlias("name")   //define field level alias
   @XStreamAsAttribute     //define field as attribute
   private String studentName;

   @XStreamImplicit        //define list as an implicit collection
   private List<Note> notes = new ArrayList<Note>();

   @XStreamOmitField       //omit a field to not to be a part of XML
   private int type;
}
```

Let us test the above annotation using XStream.

Create a java class file named XStreamTester in **C:\>XStream_WORKSPACE\com\tutorialspoint\xstream**.

File: XStreamTester.java

```
package com.tutorialspoint.xstream;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;

import java.util.ArrayList;
import java.util.List;

import javax.xml.transform.OutputKeys;
import javax.xml.transform.Source;
import javax.xml.transform.Transformer;
import javax.xml.transform.sax.SAXSource;
import javax.xml.transform.sax.SAXTransformerFactory;
import javax.xml.transform.stream.StreamResult;

import org.xml.sax.InputSource;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.annotations.XStreamAlias;
import com.thoughtworks.xstream.annotations.XStreamAsAttribute;
import com.thoughtworks.xstream.annotations.XStreamImplicit;
import com.thoughtworks.xstream.annotations.XStreamOmitField;
import com.thoughtworks.xstream.io.xml.StaxDriver;

public class XStreamTester {
   public static void main(String args[]){

      XStreamTester tester = new XStreamTester();
```

```
      XStream xstream = new XStream(new StaxDriver());
      Student student = tester.getStudentDetails();

      xstream.processAnnotations(Student.class);

      //Object to XML Conversion
      String xml = xstream.toXML(student);
      System.out.println(formatXml(xml));
   }

   private Student getStudentDetails(){

      Student student = new Student("Mahesh");

      student.addNote(new Note("first","My first assignment."));
      student.addNote(new Note("second","My Second assignment."));
      student.setType(1);

      return student;
   }

   public static String formatXml(String xml){

      try{
         Transformer serializer = SAXTransformerFactory.newInstance().newTransformer();

         serializer.setOutputProperty(OutputKeys.INDENT, "yes");
         serializer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2");

         Source xmlSource = new SAXSource(new InputSource(new
ByteArrayInputStream(xml.getBytes())));
         StreamResult res = new StreamResult(new ByteArrayOutputStream());

         serializer.transform(xmlSource, res);

         return new String(((ByteArrayOutputStream)res.getOutputStream()).toByteArray());

      }catch(Exception e){
         return xml;
      }
   }
}

@XStreamAlias("student")
class Student {

   @XStreamAlias("name")
   @XStreamAsAttribute
   private String studentName;

   @XStreamImplicit
   private List<Note> notes = new ArrayList<Note>();

   public Student(String name) {
      this.studentName = name;
   }

   public void addNote(Note note) {
      notes.add(note);
   }

   public String getName(){
      return studentName;
   }

   public List<Note> getNotes(){
      return notes;
   }
```

```
   @XStreamOmitField
   private int type;

   public int getType(){
      return type;
   }

   public void setType(int type){
      this.type = type;
   }
}

@XStreamAlias("note")
class Note {
   private String title;
   private String description;

   public Note(String title, String description) {
      this.title = title;
      this.description = description;
   }

   public String getTitle(){
      return title;
   }

   public String getDescription(){
      return description;
   }
}
```

Verify the Result

Compile the classes using **javac** compiler as follows:

```
C:\XStream_WORKSPACE\com\tutorialspoint\xstream>javac XStreamTester.java
```

Now run the XStreamTester to see the result:

```
C:\XStream_WORKSPACE\com\tutorialspoint\xstream>java XStreamTester
```

Verify the output as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<student name="Mahesh">
   <note>
      <title>first</title>
      <description>My first assignment.</description>
   </note>
   <note>
      <title>second</title>
      <description>My Second assignment.</description>
   </note>
</student>
```

In order to instruct the XStream framework to process annotation, you need to add the following command before serializing xml.

```
xstream.processAnnotations(Student.class);
```

Or

```
xstream.autodetectAnnotations(true);
```