XStream

java based library

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com


https://www.facebook.com/tutorialspointindia


https://twitter.com/tutorialspoint

About the Tutorial

XStream is a simple Java-based library to serialize Java objects to XML and vice versa.

This is a brief tutorial that adopts a simple and intuitive way to explain the basic features of XStream library and how to use them.

Audience

This tutorial has been prepared to suit the requirements of Java developers who would like to understand the basics of XStream library and use it in their Java programs.

Prerequisites

Since XStream is a Java-based library, you need to have a clear understanding of Java programming in order to make use of this library.

Copyright & Disclaimer

Table of Contents

About the Tutorial i

Audience i

Prerequisites i

Copyright & Disclaimer i

Table of Contents ii

1. OVERVIEW 1

Features 1

Common Uses 1

2. ENVIRONMENT SETUP 3

Try it Option Online 3

Local Environment Setup 3

Popular Java Editors 4

Download XStream Archive 4

Set XStream Environment 5

Set CLASSPATH Variable 5

3. FIRST APPLICATION 6

Steps to Remember 12

4. ALIASING 14

Class Aliasing 19

Field Aliasing 22

Implicit Collections Aliasing 26

Attribute Aliasing 30

Package Aliasing 33

5. ANNOTATIONS 38

6. CONVERTERS 44

Using Converter **44**

Example without Converter **45**

Example with Converter **48**

Custom Converter **52**

7. OBJECT STREAMS 58

8. WRITING JSON USING XSTREAM 62

1. OVERVIEW

XStream is a simple Java-based library to serialize Java objects to XML and vice versa.

Features

- **Easy to use** - XStream API provides a high-level facade to simplify common use cases.
- **No need to create mapping** - XStream API provides default mapping for most of the objects to be serialized.
- **Performance** - XStream is fast and is of low memory footprint, which is suitable for large object graphs or systems.
- **Clean XML** - XStream produces clean and compact XML output that is easy to read.
- **Object modification not required** - XStream serializes internal fields like private and final fields, and supports non-public and inner classes. Default constructor is not a mandatory requirement.
- **Full object graph support** - XStream allows to maintain duplicate references encountered in the object-model and also supports circular references.
- **Customizable conversion strategies** - Custom strategies can be registered in order to allow customization of a particular type to be represented as XML.
- **Security framework** - XStream provides a fair control over unmarshalled types to prevent security issues with manipulated input.
- **Error messages** - When an exception occurs due to malformed XML, it provides detailed diagnostics to fix the problem.
- **Alternative output format** - XStream supports other output formats like JSON and morphing.

Common Uses

- **Transport -** XML is a text representation of object and can be used to transport objects over the wire independent of the serialization / deserialization techniques used.
- **Persistence -** Objects can be persisted as XML in databases and can be marshalled/unmarshalled as and when required.

- **Configuration -** XML is self-explanatory and is heavily used to define configurations. Objects can also be used for configuration purpose after converting them to XML representation.
- **Unit Tests -** XStream API is JUnit compatible and can be used to enhance unit testing of application modules.

2. ENVIRONMENT SETUP

Try it Option Online

We already have set up Java Programming environment online, so that you can compile and execute all the available examples at the same time when you are doing your theory work. This gives you confidence in what you are reading and to check the result with different options. Feel free to modify any example and execute it online.

Try the following example using the **Try it** option available at the top right corner of the sample code on our website:

```
public class MyFirstJavaProgram {


    public static void main(String []args) {
       System.out.println("Hello World");
    }
}
```

For most of the examples given in this tutorial, you will find a **Try it** option in our website code sections at the top right corner that will take you to the online compiler. So just make use of it and enjoy your learning.

Local Environment Setup

If you want to set up your environment for Java programming language, then this section explains how to download and set up Java on your machine. Please follow the steps given below to set up your Java environment.

Java SE can be downloaded for free from the link:

http://www.oracle.com/technetwork/java/archive-139210.html

Follow the instructions to download Java and run the .exe to install Java on your machine. Once you have installed Java on your machine, you would need to set the environment variables to point to correct installation directories:

Setting Up the Path for Windows 2000/XP

Assuming you have installed Java in *c:\Program Files\java\jdk* directory:

1. Right-click on 'My Computer' and select 'Properties'.

2. Click the 'Environment variables' button under the 'Advanced' tab.
3. Alter the 'Path' variable so that it also contains the path to the Java executable. For example, if the path is currently set to 'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.

Setting Up the Path for Windows 95/98/ME

Assuming you have installed Java in *c:\Program Files\java\jdk* directory:

- Edit the 'C:\autoexec.bat' file and add the following line at the end: 'SET PATH=%PATH%;C:\Program Files\java\jdk\bin'

Setting Up the Path for Linux, UNIX, Solaris, FreeBSD

Environment variable PATH should be set to point to where the Java binaries have been installed. Refer to your shell documentation if you have trouble doing this.

For example, if you use *bash* as your shell, then you would add the following line at the end of your '.bashrc: export PATH=/path/to/java:$PATH'

Popular Java Editors

To write Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following:

- **Notepad**: On Windows, you can use any simple text editor like Notepad (Recommended for this tutorial) or TextPad.
- **Netbeans**: It is a Java IDE that is free and can be downloaded from http://www.netbeans.org/index.html.
- **Eclipse**: It is also a Java IDE developed by the eclipse open-source community and can be downloaded from http://www.eclipse.org/.

Download XStream Archive

Download the latest version of XStream jar file from xstream-1.4.7.jar. At the time of writing this tutorial, we have downloaded *xstream-1.4.7.jar* and copied it into C:\>XStream folder.

OS	Archive name
Windows	xstream-1.4.7.jar
Linux	xstream-1.4.7.jar
Mac	xstream-1.4.7.jar

Set XStream Environment

Set the XStream_HOME environment variable to point to the base directory location where xstream jar is stored on your machine. The following table shows how to set the XStream environment on Windows, Linux, and Mac, assuming we've extracted xstream-1.4.7.jar in the XStream folder.

OS	Description
Windows	Set the environment variable XStream_HOME to C:\XStream
Linux	export XStream_HOME=/usr/local/XStream
Mac	export XStream_HOME=/Library/XStream

Set CLASSPATH Variable

Set the CLASSPATH environment variable to point to the XStream jar location. The following table shows how to set the CLASSPATH variable on Windows, Linux, and Mac systems, assuming we've stored xstream-1.4.7.jar in the XStream folder.

OS	Description
Windows	Set the environment variable CLASSPATH to %CLASSPATH%;%XStream_HOME%\xstream-1.4.7.jar;
Linux	export CLASSPATH=$CLASSPATH:$XStream_HOME/xstream-1.4.7.jar:
Mac	export CLASSPATH=$CLASSPATH:$XStream_HOME/xstream-1.4.7.jar:

3. FIRST APPLICATION

Before going into the details of the XStream library, let us see an application in action. In this example, we've created Student and Address classes. We will create a student object and then serialize it to an XML String. Then de-serialize the same XML string to obtain the student object back.

Create a java class file named XStreamTester in C:\>XStream_WORKSPACE.

File: XStreamTester.java

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

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.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();

       //Object to XML Conversion
      String xml = xstream.toXML(student);
```

```
      System.out.println(formatXml(xml));

      //XML to Object Conversion
      Student student1 = (Student)xstream.fromXML(xml);
      System.out.println(student1);
   }

   private Student getStudentDetails(){
      Student student = new Student();
      student.setFirstName("Mahesh");
      student.setLastName("Parashar");
      student.setRollNo(1);
      student.setClassName("1st");

      Address address = new Address();
      address.setArea("H.No. 16/3, Preet Vihar.");
      address.setCity("Delhi");
      address.setState("Delhi");
      address.setCountry("India");
      address.setPincode(110012);

      student.setAddress(address);
      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}inde
         nt-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;
         }
      }
   }

class Student {
   private String firstName;
   private String lastName;
   private int rollNo;
   private String className;
   private Address address;

   public String getFirstName() {
      return firstName;
   }
   public void setFirstName(String firstName) {
      this.firstName = firstName;
   }
   public String getLastName() {
      return lastName;
   }
   public void setLastName(String lastName) {
      this.lastName = lastName;
   }
   public int getRollNo() {
      return rollNo;
```

```
   }
   public void setRollNo(int rollNo) {
      this.rollNo = rollNo;
   }
   public String getClassName() {
      return className;
   }
   public void setClassName(String className) {
      this.className = className;
   }
   public Address getAddress() {
      return address;
   }
   public void setAddress(Address address) {
      this.address = address;
   }

   public String toString(){
      StringBuilder stringBuilder = new StringBuilder();
      stringBuilder.append("Student [ ");
      stringBuilder.append("\nfirstName: ");
      stringBuilder.append(firstName);
      stringBuilder.append("\nlastName: ");
      stringBuilder.append(lastName);
      stringBuilder.append("\nrollNo: ");
      stringBuilder.append(rollNo);
      stringBuilder.append("\nclassName: ");
      stringBuilder.append(className);
      stringBuilder.append("\naddress: ");
      stringBuilder.append(address);
      stringBuilder.append(" ]");
      return stringBuilder.toString();
   }
}
```

```
class Address {
   private String area;
   private String city;
   private String state;
   private String country;
   private int pincode;

   public String getArea() {
      return area;
   }
   public void setArea(String area) {
      this.area = area;
   }
   public String getCity() {
      return city;
   }
   public void setCity(String city) {
      this.city = city;
   }
   public String getState() {
      return state;
   }
   public void setState(String state) {
      this.state = state;
   }
   public String getCountry() {
      return country;
   }
   public void setCountry(String country) {
      this.country = country;
   }
   public int getPincode() {
      return pincode;
```

```
   }
   public void setPincode(int pincode) {
      this.pincode = pincode;
   }
   public String toString(){
      StringBuilder stringBuilder = new StringBuilder();
      stringBuilder.append("\nAddress [ ");
      stringBuilder.append("\narea: ");
      stringBuilder.append(area);
      stringBuilder.append("\ncity: ");
      stringBuilder.append(city);
      stringBuilder.append("\nstate: ");
      stringBuilder.append(state);
      stringBuilder.append("\ncountry: ");
      stringBuilder.append(country);
      stringBuilder.append("\npincode: ");
      stringBuilder.append(pincode);
      stringBuilder.append(" ]");
      return stringBuilder.toString();
   }
}
```

Verify the Result

Compile the classes using javac compiler as follows:

```
C:\XStream_WORKSPACE>javac XStreamTester.java
```

Now run the XStreamTester to see the result:

```
C:\XStream_WORKSPACE>java XStreamTester
```

Verify the output as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Student>
  <firstName>Mahesh</firstName>
  <lastName>Parashar</lastName>
```

```
  <rollNo>1</rollNo>
  <className>1st</className>
  <address>
    <area>H.No. 16/3, Preet Vihar.</area>
    <city>Delhi</city>
    <state>Delhi</state>
    <country>India</country>
    <pincode>110012</pincode>
  </address>
</Student>

Student [
firstName: Mahesh
lastName: Parashar
rollNo: 1
className: 1st
address:
Address [
area: H.No. 16/3, Preet Vihar.
city: Delhi
state: Delhi
country: India
pincode: 110012 ] ]
```

Steps to Remember

Following are the important steps to be considered here.

Step 1: Create an XStream Object

Create an XStream object by passing it a StaxDriver. StaxDriver uses Stax pull parser (available from java 6) and is a fast xml parser.

```
XStream xstream = new XStream(new StaxDriver());
```

Step 2: Serialize the Object to XML

Use toXML() method to get the XML string representation of the object.

```
//Object to XML Conversion
String xml = xstream.toXML(student);
```

Step 3: De-serialize XML to Get the Object

Use fromXML() method to get the object from the XML.

```
//XML to Object Conversion
Student student1 = (Student)xstream.fromXML(xml);
```

4. ALIASING

Aliasing is a technique to customize the generated XML or to use a particular formatted XML using XStream. Let's suppose the following XML format is to be used to serialize/de-serialize the Student object.

```
<student name="Suresh">
   <note>
   <title>first</title>
      <description>My first assignment.</description>
   </note>
   <note>
      <title>second</title>
      <description>My second assignment.</description>
   </note>
</student>
```

Based on the above XML format, let's create model classes.

```
class Student {
   private String studentName;
   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;
   }
```

```
}

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;
   }
}
```

Let's test the above object's serialization 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.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();
      //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."));
      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;
         }
      }
   }

class Student {
   private String studentName;
   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;
   }
}

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"?>
<com.tutorialspoint.xstream.Student>
  <studentName>Mahesh</studentName>
  <notes>
    <com.tutorialspoint.xstream.Note>
      <title>first</title>
      <description>My first assignment.</description>
    </com.tutorialspoint.xstream.Note>
    <com.tutorialspoint.xstream.Note>
      <title>second</title>
      <description>My Second assignment.</description>
    </com.tutorialspoint.xstream.Note>
  </notes>
</com.tutorialspoint.xstream.Student>
```

In the above result, the Student object name is fully qualified. To replace it as student tag, follow the next section.

Class Aliasing

Class aliasing is used to create an alias of a fully qualified name of a class in XML. Let us modify our original example and add the following code to it.

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

Let us test the above object's serialization 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.io.xml.StaxDriver;

public class XStreamTester {
   public static void main(String args[]){
      XStreamTester tester = new XStreamTester();
      XStream xstream = new XStream(new StaxDriver());
```

```
      xstream.alias("student", Student.class);
      xstream.alias("note", Note.class);
      Student student = tester.getStudentDetails();
      //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."));
      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;
      }
   }
}
```

```
class Student {
   private String studentName;
   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;
   }
}

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>
  <studentName>Mahesh</studentName>
  <notes>
    <note>
      <title>first</title>
      <description>My first assignment.</description>
    </note>
    <note>
      <title>second</title>
      <description>My Second assignment.</description>
    </note>
  </notes>
</student>
```

In the above result, we can see that studentName is required to be renamed to name. To replace it, follow the next section.

Field Aliasing

Field aliasing is used to create an alias of a field in XML. Let us modify our example again and add the following code to it.

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

Let us test the above object's serialization 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.io.xml.StaxDriver;

public class XStreamTester {
   public static void main(String args[]){
      XStreamTester tester = new XStreamTester();
      XStream xstream = new XStream(new StaxDriver());
      xstream.alias("student", Student.class);
      xstream.alias("note", Note.class);
      xstream.aliasField("name", Student.class, "studentName");
      Student student = tester.getStudentDetails();
      //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."));
```

```
      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;
      }
   }
}

class Student {
   private String studentName;
   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;
   }
}

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</name>
  <notes>
```

```
    <note>
      <title>first</title>
      <description>My first assignment.</description>
    </note>
    <note>
      <title>second</title>
      <description>My Second assignment.</description>
    </note>
  </notes>
</student>
```

In the above result, we can see that the notes tag gets added as a list of notes. To replace it, follow the next section.

Implicit Collections Aliasing

Implicit collections aliasing is used when a collection is to be represented in XML without displaying the roots. For example, in our case, we need to display each note one by one but not in the 'notes' root node. Let us modify our example again and add the following code to it.

```
xstream.addImplicitCollection(Student.class, "notes");
```

Let us test the above object's serialization 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.io.xml.StaxDriver;

public class XStreamTester {
   public static void main(String args[]){
      XStreamTester tester = new XStreamTester();
      XStream xstream = new XStream(new StaxDriver());
      xstream.alias("student", Student.class);
      xstream.alias("note", Note.class);
      xstream.aliasField("name", Student.class, "studentName");
      xstream.addImplicitCollection(Student.class, "notes");
      Student student = tester.getStudentDetails();
      //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."));
      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;
      }
   }
}

class Student {
   private String studentName;
   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;
   }
```

```
}

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</name>
  <note>
    <title>first</title>
    <description>My first assignment.</description>
  </note>
  <note>
    <title>second</title>
```

```
    <description>My Second assignment.</description>
  </note>
</student>
```

In the above result, we can see that name is coming as a child node and we need it as an attribute of the root node. To replace it, follow the next section.

Attribute Aliasing

Attribute aliasing is used to serialize a member variable as an XML attribute. Let us modify our example again and add the following code to it.

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

Let us test the above object's serialization 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.io.xml.StaxDriver;

public class XStreamTester {
   public static void main(String args[]){
      XStreamTester tester = new XStreamTester();
      XStream xstream = new XStream(new StaxDriver());
      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");
      Student student = tester.getStudentDetails();
      //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."));
      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;
         }
      }
}

class Student {
   private String studentName;
   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;
   }
}

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>
```

Package Aliasing

Package aliasing is used to create an alias of a fully qualified name of a class in XML to a new qualified name. Let us modify our example again and change the following code.

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

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

Above code is changed as follows:

```
xstream.aliasPackage("my.company.xstream","com.tutorialspoint.xstream");
```

Let us test the above object's serialization 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.io.xml.StaxDriver;

public class XStreamTester {
   public static void main(String args[]){
      XStreamTester tester = new XStreamTester();
      XStream xstream = new XStream(new StaxDriver());
      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");
      Student student = tester.getStudentDetails();
      //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."));
      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;
      }
   }
}
```

```
class Student {
   private String studentName;
   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;
   }
}

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"?>
<my.company.xstream.Student name="Mahesh">
  <my.company.xstream.Note>
    <title>first</title>
    <description>My first assignment.</description>
  </my.company.xstream.Note>
  <my.company.xstream.Note>
    <title>second</title>
    <description>My Second assignment.</description>
  </my.company.xstream.Note>
</my.company.xstream.Student>
```

5. 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);
```

6. CONVERTERS

XStream converters are the key components of the XStream library, which are responsible to convert an object to XML and vice versa. XStream provides numerous converters for common types such as primitives, String, File, Collections, arrays, and Dates.

Using Converter

Let us use a SingleValueConvertor whose purpose is to convert an object into a single string. We will use SingleValueConvertor to write an object as attribute string.

Create a Converter

```
class NameConverter implements SingleValueConverter  {

   public Object fromString(String name) {
      String[] nameparts = name.split(",");
      return new Name(nameparts[0], nameparts[1]);
   }

   public String toString(Object name) {
      return ((Name)name).getFirstName() + "," +
      ((Name)name).getLastName();
   }

   public boolean canConvert(Class type) {
      return type.equals(Name.class);
   }
}
```

Register a Converter

```
xstream.registerConverter(new NameConverter());
```

Example without Converter

Let us first test the code without converter in 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 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.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.autodetectAnnotations(true);
           //Object to XML Conversion
           String xml = xstream.toXML(student);
           System.out.println(formatXml(xml));
```

```
    }

    private Student getStudentDetails(){
          Student student = new Student("Mahesh","Parashar");
          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 Name studentName;
```

```
   public Student(String firstName, String lastName) {
      this.studentName = new Name(firstName, lastName);
   }

   public Name getName(){
      return studentName;
   }
}

class Name {
   private String firstName;
   private String lastName;

   public Name(String firstName, String lastName) {
      this.firstName = firstName;
      this.lastName = lastName;
   }

   public String getFirstName(){
      return firstName;
   }

   public String getLastName(){
      return lastName;
   }
}
```

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>
    <firstName>Mahesh</firstName>
    <lastName>Parashar</lastName>
  </name>
</student>
```

Example with Converter

Let us now test the code with converter in 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 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.converters.SingleValueConverter;
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.autodetectAnnotations(true);
      xstream.registerConverter(new NameConverter());
      //Object to XML Conversion
      String xml = xstream.toXML(student);
      System.out.println(formatXml(xml));
   }

   private Student getStudentDetails(){
      Student student = new Student("Mahesh","Parashar");
      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 Name studentName;

   public Student(String firstName, String lastName) {
      this.studentName = new Name(firstName, lastName);
   }

   public Name getName(){
      return studentName;
   }
}

class Name {
   private String firstName;
   private String lastName;

   public Name(String firstName, String lastName) {
      this.firstName = firstName;
      this.lastName = lastName;
   }

   public String getFirstName(){
      return firstName;
   }
```

```
   public String getLastName(){
      return lastName;
   }
}

class NameConverter implements SingleValueConverter  {

   public Object fromString(String name) {
      String[] nameparts = name.split(",");
      return new Name(nameparts[0], nameparts[1]);
   }

   public String toString(Object name) {
      return ((Name)name).getFirstName() + "," +
      ((Name)name).getLastName();
   }

   public boolean canConvert(Class type) {
      return type.equals(Name.class);
   }
}
```

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,Parashar"/>
```

Custom Converter

XStream allows writing a converter from scratch, so that the developer can write a completely new implementation on how to serialize an object to XML and vice versa. A converter interface provides three methods:

- canConvert – It is a check for supported object type serialization.
- marshal – It serializes an object to XML.
- unmarshal – It de-serializes an object from XML.

Step 1: Implement Converter Interface

```
class StudentConverter implements Converter {

   public boolean canConvert(Class object) {
      return object.equals(Student.class);
   }
   public void marshal(Object value, HierarchicalStreamWriter writer,
      MarshallingContext context) {
      Student student = (Student) value;
      writer.startNode("name");
      writer.setValue(student.getName().getFirstName() + "," +
      student.getName().getLastName());
      writer.endNode();
   }

   public Object unmarshal(HierarchicalStreamReader reader,
      UnmarshallingContext context) {
      reader.moveDown();
      String[] nameparts = reader.getValue().split(",");
      Student student = new Student(nameparts[0],nameparts[1]);
      reader.moveUp();
      return student;
```

```
   }
}
```

Step 2: Register Converter

```
xstream.registerConverter(new StudentConverter());
```

Let us now test the code with converter in 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 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.converters.Converter;
import com.thoughtworks.xstream.converters.MarshallingContext;
import com.thoughtworks.xstream.converters.UnmarshallingContext;
import com.thoughtworks.xstream.io.HierarchicalStreamReader;
import com.thoughtworks.xstream.io.HierarchicalStreamWriter;
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.autodetectAnnotations(true);
      xstream.registerConverter(new StudentConverter());
      //Object to XML Conversion
      String xml = xstream.toXML(student);
      System.out.println(formatXml(xml));
   }

   private Student getStudentDetails(){
      Student student = new Student("Mahesh","Parashar");
      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")
   private Name studentName;

   public Student(String firstName, String lastName) {
      this.studentName = new Name(firstName, lastName);
   }

   public Name getName(){
      return studentName;
   }
}

class Name {
   private String firstName;
   private String lastName;

   public Name(String firstName, String lastName) {
      this.firstName = firstName;
      this.lastName = lastName;
   }

   public String getFirstName(){
      return firstName;
   }
   public String getLastName(){
      return lastName;
   }
}
```

```
class StudentConverter implements Converter {

   public void marshal(Object value, HierarchicalStreamWriter writer,
      MarshallingContext context) {
      Student student = (Student) value;
      writer.startNode("name");
      writer.setValue(student.getName().getFirstName() + "," +
      student.getName().getLastName());
      writer.endNode();
   }
   public Object unmarshal(HierarchicalStreamReader reader,
      UnmarshallingContext context) {
      reader.moveDown();
      String[] nameparts = reader.getValue().split(",");
      Student student = new Student(nameparts[0],nameparts[1]);
      reader.moveUp();
      return student;
   }
   public boolean canConvert(Class object) {
      return object.equals(Student.class);
   }
}
```

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,Parashar</name>
</student>
```

7. OBJECT STREAMS

XStream provides alternative implementations of java.io.ObjectInputStream and java.io.ObjectOutputStream so that streams of objects can be serialized or de-serialized from XML. This is particularly useful when large sets of objects are to be processed, keeping one object in memory at a time.

Syntax: reateObjectOutputStream()

```
ObjectOutputStream objectOutputStream =
xstream.createObjectOutputStream(new FileOutputStream("test.txt"));
```

Syntax: createObjectInputStream()

```
ObjectInputStream objectInputStream =
xstream.createObjectInputStream(new FileInputStream("test.txt"));
```

Let us now test the code with object streams in 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.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.annotations.XStreamAlias;
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());
xstream.autodetectAnnotations(true);

Student student1 = new Student("Mahesh","Parashar");
Student student2 = new Student("Suresh","Kalra");
Student student3 = new Student("Ramesh","Kumar");
Student student4 = new Student("Naresh","Sharma");

try {
   ObjectOutputStream objectOutputStream =
   xstream.createObjectOutputStream(new FileOutputStream("test.txt"));
   objectOutputStream.writeObject(student1);
   objectOutputStream.writeObject(student2);
   objectOutputStream.writeObject(student3);
   objectOutputStream.writeObject(student4);
   objectOutputStream.writeObject("Hello World");
   objectOutputStream.close();

   ObjectInputStream objectInputStream =
   xstream.createObjectInputStream(new FileInputStream("test.txt"));
   Student student5 = (Student)objectInputStream.readObject();
   Student student6 = (Student)objectInputStream.readObject();
   Student student7 = (Student)objectInputStream.readObject();
   Student student8 = (Student)objectInputStream.readObject();
   String text = (String)objectInputStream.readObject();
   System.out.println(student5);
   System.out.println(student6);
   System.out.println(student7);
   System.out.println(student8);
   System.out.println(text);

} catch (IOException e) {
   e.printStackTrace();
} catch (ClassNotFoundException e) {
```

```
         e.printStackTrace();
      }
   }
}

@XStreamAlias("student")
class Student {

   private String firstName;
   private String lastName;

   public Student(String firstName, String lastName) {
      this.firstName = firstName;
      this.lastName = lastName;
   }

   public String getFirstName() {
      return firstName;
   }

   public String getLastName() {
      return lastName;
   }

   public String toString(){
      return "Student [ firstName: "+firstName+", lastName: "+ lastName+ " ]";
   }
}
```

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:

```
Student [ firstName: Mahesh, lastName: Parashar ]
Student [ firstName: Suresh, lastName: Kalra ]
Student [ firstName: Ramesh, lastName: Kumar ]
Student [ firstName: Naresh, lastName: Sharma ]
Hello World
```

Look at the content of the test.txt present at C:\>XStream_WORKSPACE\com\tutorialspoint\xstream folder.

```
<?xml version="1.0" ?>
<object-stream>
   <student>
      <firstName>Mahesh</firstName>
      <lastName>Parashar</lastName>
   </student>
   <student>
      <firstName>Suresh</firstName>
      <lastName>Kalra</lastName>
   </student>
   <student>
      <firstName>Ramesh</firstName>
      <lastName>Kumar</lastName>
   </student>
   <student>
      <firstName>Naresh</firstName>
      <lastName>Sharma</lastName>
   </student>
   <string>Hello World</string>
</object-stream>
```

8. WRITING JSON USING XSTREAM

XStream supports JSON by initializing XStream object with an appropriate driver. XStream currently supports JettisonMappedXmlDriver and JsonHierarchicalStreamDriver.

Let us now test the code with json handling in 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.Writer;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.annotations.XStreamAlias;
import com.thoughtworks.xstream.io.HierarchicalStreamWriter;
import com.thoughtworks.xstream.io.json.JsonHierarchicalStreamDriver;
import com.thoughtworks.xstream.io.json.JsonWriter;

public class XStreamTester {
   public static void main(String args[]){
      XStreamTester tester = new XStreamTester();
      XStream xstream = new XStream(new JsonHierarchicalStreamDriver() {
         public HierarchicalStreamWriter createWriter(Writer writer) {
            return new JsonWriter(writer, JsonWriter.DROP_ROOT_MODE);
            }
      });

      Student student = new Student("Mahesh","Parashar");
      xstream.setMode(XStream.NO_REFERENCES);
      xstream.alias("student", Student.class);
```

```
        System.out.println(xstream.toXML(student));

     }
}

@XStreamAlias("student")
class Student {

     private String firstName;
     private String lastName;

     public Student(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
     }

     public String getFirstName() {
        return firstName;
     }

     public String getLastName() {
        return lastName;
     }

     public String toString(){
        return "Student [ firstName: "+firstName+", lastName: "+ lastName+ " ]";
     }
}
```

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:

```
{
  "firstName": "Mahesh",
  "lastName": "Parashar"
}
```