L

**You said**

Share XML Transformation Example using Jaxon library using XSLT compatible withJava6

**Copilot said**

Here's an example of how to perform XML transformation using the Jaxon library with XSLT in Java 6:

First, ensure that you have the Jaxon library added to your project. You can download it from here.

Now, here's a sample Java code to demonstrate the transformation:

java

Copy

import net.sf.saxon.TransformerFactoryImpl;

import javax.xml.transform.\*;

import javax.xml.transform.stream.StreamResult;

import javax.xml.transform.stream.StreamSource;

import java.io.File;

public class XMLTransformExample {

public static void main(String[] args) {

try {

// Instantiate a TransformerFactory

TransformerFactory factory = new TransformerFactoryImpl();

// Load the XSLT file

Source xslt = new StreamSource(new File("transform.xslt"));

// Compile the stylesheet

Transformer transformer = factory.newTransformer(xslt);

// Load the input XML document

Source xml = new StreamSource(new File("input.xml"));

// Set the output

Result output = new StreamResult(new File("output.xml"));

// Perform the transformation

transformer.transform(xml, output);

System.out.println("Transformation completed successfully.");

} catch (TransformerException e) {

e.printStackTrace();

}

}

}

Here's a brief explanation of the code:

* The TransformerFactoryImpl from the Jaxon library is used to create a TransformerFactory.
* The XSLT file (transform.xslt) and the input XML file (input.xml) are loaded using StreamSource.
* The Transformer object is created by compiling the XSLT stylesheet.
* The transformation is performed by calling the transform method, which takes the input XML and the output file as parameters.

Make sure you have transform.xslt and input.xml in the same directory as your Java code, or provide the correct path to these files.

Let me know if you need further assistance or if there's anything else you'd like to know!

**You said**

https://jaxen.codehaus.org not working any alternative

**Copilot said**

No worries! If the Jaxon library isn't working for you, you can try using the **Saxon** library for XML transformations. It's a robust and widely-used library for XSLT transformations and is compatible with Java 6.

Here's an example of how to use Saxon for XML transformation:

1. **Add Saxon to your project**: You can download it from the Saxonica website or include it as a Maven dependency:

xml

Copy

<dependency>

<groupId>net.sf.saxon</groupId>

<artifactId>Saxon-HE</artifactId>

<version>9.9.1-7</version>

</dependency>

1. **Sample Java code**:

java

Copy

import javax.xml.transform.\*;

import javax.xml.transform.stream.StreamResult;

import javax.xml.transform.stream.StreamSource;

import java.io.File;

public class XMLTransformExample {

public static void main(String[] args) {

try {

// Load the XSLT file

Source xslt = new StreamSource(new File("transform.xslt"));

// Load the input XML document

Source xml = new StreamSource(new File("input.xml"));

// Set the output

Result output = new StreamResult(new File("output.xml"));

// Instantiate a TransformerFactory

TransformerFactory factory = new net.sf.saxon.TransformerFactoryImpl();

// Compile the stylesheet

Transformer transformer = factory.newTransformer(xslt);

// Perform the transformation

transformer.transform(xml, output);

System.out.println("Transformation completed successfully.");

} catch (TransformerException e) {

e.printStackTrace();

}

}

}

This code is similar to the previous example but uses the Saxon library instead of Jaxon. Make sure you have transform.xslt and input.xml in the same directory as your Java code, or provide the correct path to these files.

Does this help? Let me know if you have any questions or need further assistance!

**You said**

One of Application is using Xalan to do XML Transformation and having problem transforming XML having records More than 62000 , with No more DTTMID available. Looking for Alternative of using Jaxon Library . summarixe teh changes required to use jaxon library and is there any such limitation.

**Copilot said**

To switch from Xalan to the Jaxon library for XML transformation, you'll need to make a few changes to your code. Here's a summary of the steps:

1. **Add Jaxon Dependency**: Ensure you have the Jaxon library added to your project. You can include it as a Maven dependency:

xml

Copy

<dependency>

<groupId>net.sf.saxon</groupId>

<artifactId>Saxon-HE</artifactId>

<version>9.9.1-7</version>

</dependency>

1. **Update Your Code**: Replace the Xalan-specific code with Jaxon-specific code. Here's an example:

java

Copy

import javax.xml.transform.\*;

import javax.xml.transform.stream.StreamResult;

import javax.xml.transform.stream.StreamSource;

import java.io.File;

public class XMLTransformExample {

public static void main(String[] args) {

try {

// Load the XSLT file

Source xslt = new StreamSource(new File("transform.xslt"));

// Load the input XML document

Source xml = new StreamSource(new File("input.xml"));

// Set the output

Result output = new StreamResult(new File("output.xml"));

// Instantiate a TransformerFactory

TransformerFactory factory = new net.sf.saxon.TransformerFactoryImpl();

// Compile the stylesheet

Transformer transformer = factory.newTransformer(xslt);

// Perform the transformation

transformer.transform(xml, output);

System.out.println("Transformation completed successfully.");

} catch (TransformerException e) {

e.printStackTrace();

}

}

}

1. **Check for Limitations**: The Jaxon library should handle large XML files without the same limitations as Xalan. However, it's always a good idea to test with your specific use case to ensure performance and correctness.

Would you like more detailed guidance on any of these steps?

**You said**

Yes , what kind of application changes. would it be possibel to manage with Import changes or require Application Code changes

**Copilot said**

Transitioning from Xalan to Jaxon (or more specifically, Saxon as Jaxon is no longer available) will require some changes, but the good news is that a lot of it can be managed through import changes and minor adjustments in your application code. Here's a detailed breakdown:

**Changes Needed**

1. **Dependencies**:
   * **Remove Xalan Dependency**: Ensure that any Xalan-specific libraries are removed from your project.
   * **Add Saxon Dependency**:

xml

Copy

<dependency>

<groupId>net.sf.saxon</groupId>

<artifactId>Saxon-HE</artifactId>

<version>9.9.1-7</version>

</dependency>

1. **Import Statements**:
   * Update your import statements to use Saxon instead of Xalan.

**Before (Xalan)**:

java

Copy

import org.apache.xalan.xsltc.trax.TransformerFactoryImpl;

**After (Saxon)**:

java

Copy

import net.sf.saxon.TransformerFactoryImpl;

1. **Transformer Factory Initialization**:
   * Change the way you instantiate the TransformerFactory.

**Before (Xalan)**:

java

Copy

TransformerFactory factory = new org.apache.xalan.xsltc.trax.TransformerFactoryImpl();

**After (Saxon)**:

java

Copy

TransformerFactory factory = new net.sf.saxon.TransformerFactoryImpl();

1. **General Application Code**:
   * Ensure your XML and XSLT file loading mechanisms are compatible with the Saxon library. Typically, this part does not need major changes if you're using standard Java libraries for file handling.

**Example Transformation Code**:

java

Copy

import javax.xml.transform.\*;

import javax.xml.transform.stream.StreamResult;

import javax.xml.transform.stream.StreamSource;

import java.io.File;

public class XMLTransformExample {

public static void main(String[] args) {

try {

// Load the XSLT file

Source xslt = new StreamSource(new File("transform.xslt"));

// Load the input XML document

Source xml = new StreamSource(new File("input.xml"));

// Set the output

Result output = new StreamResult(new File("output.xml"));

// Instantiate a TransformerFactory

TransformerFactory factory = new net.sf.saxon.TransformerFactoryImpl();

// Compile the stylesheet

Transformer transformer = factory.newTransformer(xslt);

// Perform the transformation

transformer.transform(xml, output);

System.out.println("Transformation completed successfully.");

} catch (TransformerException e) {

e.printStackTrace();

}

}

}

**Limitations**

Saxon does not have the same limitation as Xalan regarding large XML files. However, performance can vary based on the complexity of your transformations and the hardware resources available. Here are a few tips:

* **Memory Management**: Ensure your application has sufficient heap memory allocated, especially when handling large XML files.
* **Performance Testing**: It's crucial to test your application thoroughly to identify any performance bottlenecks and optimize accordingly.

By making these adjustments, you should be able to migrate from Xalan to Saxon without major disruptions to your existing codebase.

Feel free to ask if you need more specifics or run into any issues!