XQUERY - ENVIRONMENT SETUP

This chapter elaborates how to set up XQuery library in a local development environment.

We are using an open source standalone XQuery processor Saxon Home Edition *Saxon − HE* which is widely used. This processor supports XSLT 2.0, XQuery 3.0, and XPath 3.0 and is highly optimized for performance. Saxon XQuery processor can be used without having any XML database. We'll use a simple XML document as our database in our examples.

In order to use Saxon XQuery processor, you should have saxon9he.jar, saxon9-test.jar, saxon9-unpack, saxon9-xqj.jar in your application's classpath. These jar files are available in the download file **SaxonHE9-6-0-1J.zip** Download SaxonHE9-6-0-1J.zip.

Example

We'll use the Java-based Saxon XQuery processor to test books.xqy, a file containing XQuery expression against our sample XML document, i.e., books.xml.

In this example, we'll see how to write and process a query to get the title elements of the books whose price is greater than 30.

books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<books>

   <book category="JAVA">
      <title lang="en">Learn Java in 24 Hours</title>
      <author>Robert</author>
      <year>2005</year>
      <price>30.00</price>
   </book>

   <book category="DOTNET">
      <title lang="en">Learn .Net in 24 hours</title>
      <author>Peter</author>
      <year>2011</year>
      <price>40.50</price>
   </book>

   <book category="XML">
      <title lang="en">Learn XQuery in 24 hours</title>
      <author>Robert</author>
      <author>Peter</author>
      <year>2013</year>
      <price>50.00</price>
   </book>

   <book category="XML">
      <title lang="en">Learn XPath in 24 hours</title>
      <author>Jay Ban</author>
      <year>2010</year>
      <price>16.50</price>
   </book>

</books>
```

books.xqy

```
for $x in doc("books.xml")/books/book
where $x/price>30
return $x/title
```

XQueryTester.java

```
package com.tutorialspoint.xquery;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

import javax.xml.xquery.XQConnection;
import javax.xml.xquery.XQDataSource;
import javax.xml.xquery.XQException;
import javax.xml.xquery.XQPreparedExpression;
import javax.xml.xquery.XQResultSequence;

import com.saxonica.xqj.SaxonXQDataSource;

public class XQueryTester {
   public static void main(String[] args){
      try {
         execute();
      }

      catch (FileNotFoundException e) {
         e.printStackTrace();
      }

      catch (XQException e) {
         e.printStackTrace();
      }
   }

   private static void execute() throws FileNotFoundException, XQException{
      InputStream inputStream = new FileInputStream(new File("books.xqy"));
      XQDataSource ds = new SaxonXQDataSource();
      XQConnection conn = ds.getConnection();
      XQPreparedExpression exp = conn.prepareExpression(inputStream);
      XQResultSequence result = exp.executeQuery();

      while (result.next()) {
         System.out.println(result.getItemAsString(null));
      }
   }
}
```

Steps to Execute XQuery against XML

- **Step 1** − Copy XQueryTester.java to any location, say, **E:** > **java**
- **Step 2** − Copy books.xml to the same location, **E:** > **java**
- **Step 3** − Copy books.xqy to the same location, **E:** > **java**
- **Step 4** − Compile XQueryTester.java using console. Make sure you have JDK 1.5 or later installed on your machine and classpaths are configured. For details on how to use JAVA, see our JAVA Tutorial

```
E:\java\javac XQueryTester.java
```

- **Step 5** − Execute XQueryTester

```
E:\java\java XQueryTester
```

Output

You'll get the following result −

```
<title lang="en">Learn .Net in 24 hours</title>
<title lang="en">Learn XQuery in 24 hours</title>
```

Understanding Example

- books.xml represents the sample data.
- books.xqy represents the XQuery expression which is to be executed on books.xml. We'll understand the expression in details in next chapter.
- XQueryTester, a Java-based XQuery executor program, reads the books.xqy, passes it to the XQuery expression processor, and executes the expression. Then the result is printed.