

# XML Project

Ruixuan Zhang

# Catalog

<b>Catalog.....</b>	<b>2</b>
<b>1. Project Summary.....</b>	<b>3</b>
1.1. Project Overview.....	3
1.2. Project Objective .....	3
1.3. Detailed Steps.....	3
<b>2. XQuery and Java Files .....</b>	<b>5</b>
2.1. Files overview .....	5
2.2. Java Source Code .....	6
2.2.1 XQueryTest.java .....	6
2.2.2 FileUtils.java .....	7
2.2.3 Dom_Parser.java .....	8
2.3. XQuery and Output Files.....	9
2.3.1 xquery.txt.....	9
2.3.2 results.xml .....	10
2.3.2 results.xml .....	11
<b>3. VoiceXML Related Files.....</b>	<b>12</b>
3.1. Files overview .....	12
3.2. PHP Source Code .....	13
3.2.1 main.php.....	13
3.2.2 control.php .....	14
3.3. XSLT Files.....	15
3.3.1 main.xslt .....	15
3.3.2 Five Secondary Xslt Files.....	16
<b>4. Conclusion .....</b>	<b>21</b>

# 1. Project Summary

## 1.1. Project Overview

The Project is based on the large data set “mondial.xml”, then use several XML related technologies, namely, XQuery, Java Dom, Xslt and VoiceXML, to apply an information query application, realize a simple management over a large XML data document.

## 1.2. Project Objective

Use XQuery extra the top 5 cities with the biggest population, then find which 5 countries contain these 5 cities and find out the inflation, government, overall GDP and land area per capita of these 5 countries.

Then write the result into an XML file, after that, based on Java Dom parsing technology, parse the result XML file, and write the result into a plain text file.

Finally, based on the XML result file, take advantage of PHP server code and Xslt, make an interactive VoiceXML application. When user call the phone number (415) 612-1346, follow the instruction, input the country, then the phone will answer the data information about that country.

## 1.3. Detailed Steps

First of all, based on Java Saxon XQuery API, write the XQuery expression in a txt file called “xquery.txt”, take advantage of Java I/O, input the XQuery expression into Java program, let Saxon API parse the XQuery and original “mondial.xml” file. After that, the Java program returns out expected output,

print them in console as well as write them into “results.xml” file.

The second step is using Java Dom parsing program, read “results.xml” file, and change the format of it into a plain text format, then write the output into “output.txt” file.

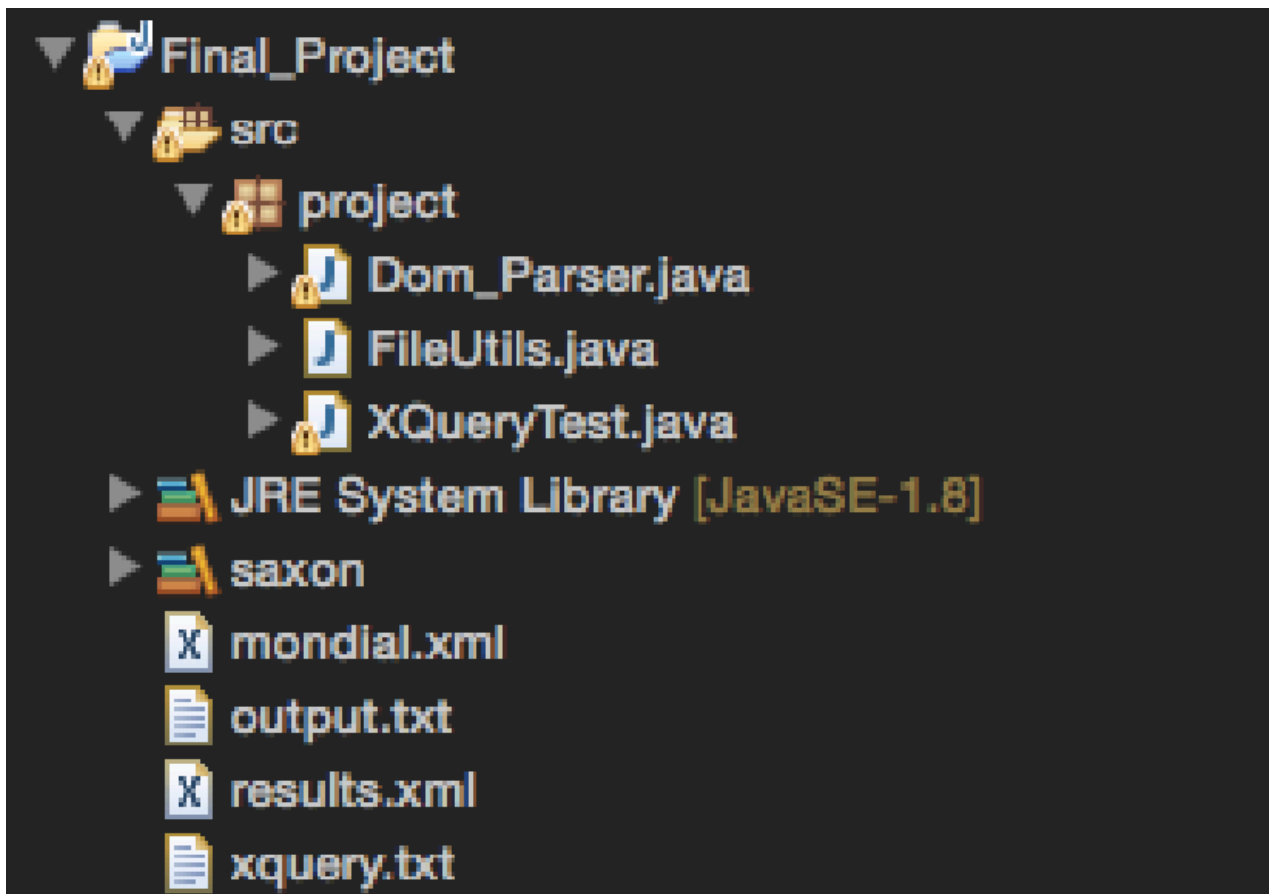
The third step, depend upon the “results.xml” file, which contains several data information about five target countries, write PHP, Xslt document and upload them into SMU Lyle server, the website url of the file folder is [lyle.smu.edu/~ruixuanz/cse7347/rush/project/](http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/), the VoiceXML index file is: [lyle.smu.edu/~ruixuanz/cse7347/rush/project/main.php](http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/main.php). After binding the Voxeo account to the index file, then it would translate “results.xml” into a VoiceXML application. After dial the phone number (415) 612-1346, after inputting the country, then the program will return the data about that country’s inflation rate(in percentage), government, overall GDP(in USD) and its land area per capita(in m<sup>2</sup>/person) to user.

## 2. XQuery and Java Files

### 2.1. Files overview

“monidal.xml” is the original XML data set, XQuery expression is inside the file “output.txt”.

There are 3 sources Java files. “XQueryTest.java” is used to parse the “mondial.xml” by “xquery.txt”, with the help of “FileUtils.java”, write the XQuery result into “results.xml”. Then “Dom\_Parser.java” could read the “result.xml”, parse it and change the format, write the output into “output.txt”.



## 2.2. Java Source Code

### 2.2.1 XQueryTest.java

```
22
23 public class XQueryTest {
24
25     public XQueryTest() {
26     }
27
28     public static void main (String [] args) throws IOException {
29         StringBuffer sb = new StringBuffer();
30         FileUtils.readToBuffer(sb, "xquery.txt");
31         System.out.println("Your XQuery expression is:\n"+sb);
32         Processor proc = new Processor(false);
33         DocumentBuilder docbuilder = proc.newDocumentBuilder();
34         XPathCompiler xpathCompiler = proc.newXPathCompiler();
35         XQueryCompiler xqueryCompiler = proc.newXQueryCompiler();
36         XsltCompiler xsltCompiler = proc.newXsltCompiler();
37         try {
38             Configuration config = new Configuration();
39             StaticQueryContext sqc = new StaticQueryContext(config);
40             XQueryExpression xqe = sqc.compileQuery(sb.toString());
41             DynamicQueryContext dqc = new DynamicQueryContext(config);
42             dqc.setContextItem(config.buildDocument(new StreamSource("mondial.xml")));
43             Properties props = new Properties();
44             xqe.run(dqc, new StreamResult(new File("results.xml")), props);
45             ByteArrayOutputStream bos = new ByteArrayOutputStream();
46             StreamResult sr = new StreamResult(bos);
47             xqe.run(dqc, sr, props);
48             System.out.println("SAXON Time Data written to file");
49             System.out.println("SAXON: \n" + bos.toString() );
50         }
51         catch (XPathException e1) {
52             System.out.println("Saxon XPATH Exception thrown." + e1);
53         }
54     }
55 }
56
```

### 2.2.2 FileUtils.java

```
1 package project;
2 import java.io.BufferedReader;
3 import java.io.FileInputStream;
4 import java.io.IOException;
5 import java.io.InputStream;
6 import java.io.InputStreamReader;
7
8
9 public class FileUtils {
10
11     public static void readToBuffer(StringBuffer buffer, String filePath) throws IOException {
12         InputStream is = new FileInputStream(filePath);
13         String line;
14         BufferedReader reader = new BufferedReader(new InputStreamReader(is));
15         line = reader.readLine();
16         while (line != null) {
17             buffer.append(line);
18             buffer.append("\n");
19             line = reader.readLine();
20         }
21         reader.close();
22         is.close();
23     }
24
25
26     public static String readFile(String filePath) throws IOException {
27         StringBuffer sb = new StringBuffer();
28         FileUtils.readToBuffer(sb, filePath);
29         return sb.toString();
30     }
31 }
```

### 2.2.3 Dom\_Parser.java

```
6 public class Dom_Parser
7 {
8     public static void main(String args[]) throws IOException
9     {
10         GiveData give=new GiveData();
11         String str1="***This txt file translate the 'result.xml' into plain text***\n\n";
12         try
13         {
14             give.sb.append(str1);
15             DocumentBuilderFactory factory=DocumentBuilderFactory.newInstance();
16             DocumentBuilder domParser=factory.newDocumentBuilder();
17             Document document=domParser.parse(new File("results.xml"));
18             NodeList nodeList=document.getChildNodes();
19             give.output(nodeList);
20             String out=give.sb.toString();
21             File txt=new File("output.txt");
22             if(!txt.exists()){
23                 txt.createNewFile();
24             }
25             byte bytes[]=new byte[1024];
26             bytes=out.getBytes();
27             int b=out.length();
28             FileOutputStream fos=new FileOutputStream(txt);
29             fos.write(bytes,0,b);
30             fos.close();
31         }
32         catch(Exception e)
33         {
34             System.out.println(e);
35         }
36     }
37 }
38 class GiveData
39 {
40     double average=0,m=0;
41     static StringBuffer sb=new StringBuffer("");
42     public void output(NodeList nodeList)//recursion
43     {
44         int size=nodeList.getLength();
45         for(int k=0;k<size;k++)
46         {
47             Node node=nodeList.item(k);
48             if(node.getNodeType()==Node.TEXT_NODE){
49                 Text textNode=(Text)node;
50                 String content=textNode.getWholeText();
51                 System.out.print(content);
52                 sb.append(content);
53                 Element parent=(Element)textNode.getParentNode();
54                 boolean boo=(parent.getNodeName().equals("price"));
55                 if(boo==true)
56                 {
57                     //
58                     content=textNode.getWholeText();
59                     average=average+Double.parseDouble(content.trim());
60                     m++;
61                 }
62             }
63             if(node.getNodeType()==Node.ELEMENT_NODE){
64                 Element elementNode=(Element)node;
65                 String name=elementNode.getNodeName();
66                 System.out.print(name+":");
67                 sb.append(name+":");
68                 NodeList nodes=elementNode.getChildNodes();
69                 output(nodes);
70             }
71         }
72     }
73 }
```



## 2.3. XQuery and Output Files

### 2.3.1 xquery.txt

```
1 declare namespace functx = "http://www.functx.com";
2
3
4 declare function local:land_area_per_capita_number(
5   $total_area as xs:decimal?,
6   $population as xs:decimal?) as xs:decimal?
7 {
8   let $lapc := round-half-to-even($total_area * 1000000 div $population,2)
9   return $lapc
10 };
11
12
13 declare function functx:trim( $arg as xs:string? ) as xs:string {
14   replace(replace($arg, '\s+', ''), '^\\s+', '')
15 } ;
16
17
18
19 element myData{
20   let $cities := doc("mondial.xml")//city
21
22   let $target_city_id := (let $cities := doc("mondial.xml")//city
23
24     for $population in $cities/population
25
26     let $city_name := data($population/../name[1])
27
28     let $city_id := data($population/../@id)
29
30     order by xs:decimal($population) descending
31
32     return ($city_id) )[position() = 1 to 5]
33
34   for $c_id in $cities/@id
35
36   for $target_id in $target_city_id
37
38   let $numresult:=local:land_area_per_capita_number(data($c_id/ancestor::country/@total_area),
39     data($c_id/ancestor::country/@population))
40
41   where $c_id=$target_id
42
43   order by $numresult descending
44
45   return element country {(
46     element name {functx:trim($c_id/ancestor::country/name[1])},
47     element inflation {string($c_id/ancestor::country/@inflation)},
48     element government {string($c_id/ancestor::country/@government)},
49     element lapc {string($numresult)},
50     element gdp {string($c_id/ancestor::country/@gdp_total)}}
51 }
52
```

### 2.3.2 results.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <myData>
3   <country>
4     <name>Brazil</name>
5     <inflation>23</inflation>
6     <government>federal republic</government>
7     <lapc>52329.41</lapc>
8     <gdp>976800</gdp>
9   </country>
10  <country>
11    <name>Mexico</name>
12    <inflation>52</inflation>
13    <government>federal republic operating under a centralized government</government>
14    <lapc>20596.21</lapc>
15    <gdp>721400</gdp>
16  </country>
17  <country>
18    <name>Pakistan</name>
19    <inflation>13</inflation>
20    <government>republic</government>
21    <lapc>6218.8</lapc>
22    <gdp>274200</gdp>
23  </country>
24  <country>
25    <name>India</name>
26    <inflation>9</inflation>
27    <government>federal republic</government>
28    <lapc>3452.96</lapc>
29    <gdp>1408700</gdp>
30  </country>
31  <country>
32    <name>Korea</name>
33    <inflation>4.3</inflation>
34    <government>republic</government>
35    <lapc>2165.24</lapc>
36    <gdp>590700</gdp>
37  </country>
38 </myData>
```

### 2.3.2 results.xml

```
1  ***This txt file translate the 'result.xml' into plain text***
2
3  myData:
4      country:
5          name:Brazil
6          inflation:23
7          government:federal republic
8          lapc:52329.41
9          gdp:976800
10
11     country:
12         name:Mexico
13         inflation:52
14         government:federal republic operating under a centralized government
15         lapc:20596.21
16         gdp:721400
17
18     country:
19         name:Pakistan
20         inflation:13
21         government:republic
22         lapc:6218.8
23         gdp:274200
24
25     country:
26         name:India
27         inflation:9
28         government:federal republic
29         lapc:3452.96
30         gdp:1408700
31
32     country:
33         name:Korea
34         inflation:4.3
35         government:republic
36         lapc:2165.24
37         gdp:590700
38
39
```

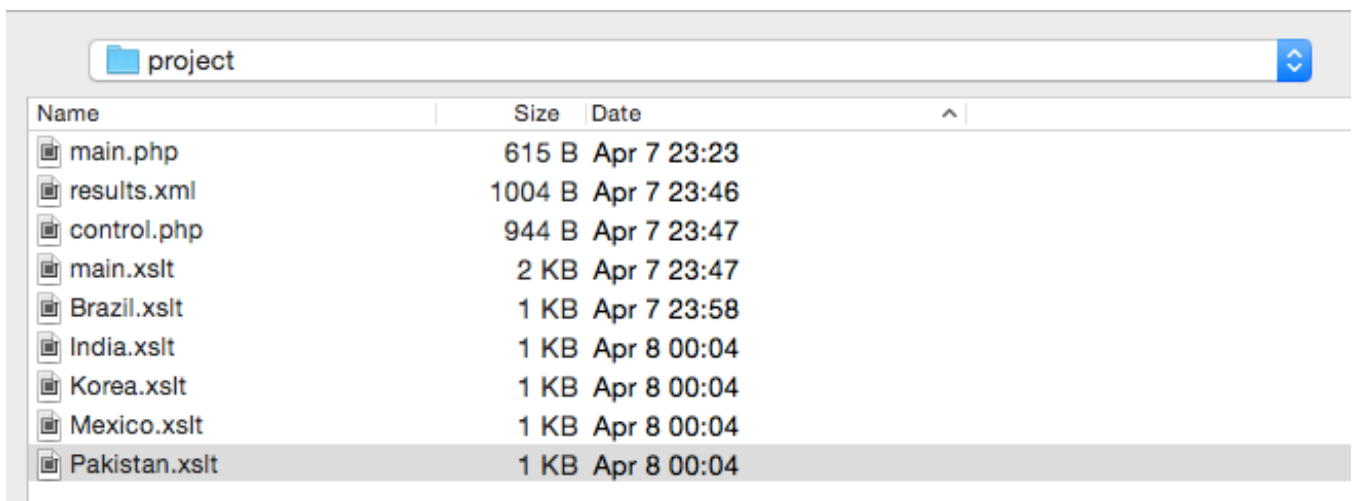
## 3. VoiceXML Related Files

### 3.1. Files overview

This VoiceXML application is mainly based upon the previous XQuery output “results.xml”. These files has uploaded to the SMU Lyle server, the website url of the file folder is [lyle.smu.edu/~ruixuanz/cse7347/rush/project/](http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/).

The Voxeo account has connected to “main.php”, whenever call the phone, this php file is executed, it would integrate the “results.xml” and “main.xslt” into a vxml file, prompt the main menu choice through voice to the user.

After user choose a certain country, a parameter would be submitted to “control.php” file, this php file could decide which one of five xslt files (“Brazil.xslt”, “India.xslt”, “Korea.xslt”, “Mexico.xslt”, “Palistan.xslt”) should be called in the succeeding process.



The screenshot shows a web browser interface displaying a file directory for a folder named 'project'. The directory listing includes columns for 'Name', 'Size', and 'Date'. The files listed are: main.php (615 B, Apr 7 23:23), results.xml (1004 B, Apr 7 23:46), control.php (944 B, Apr 7 23:47), main.xslt (2 KB, Apr 7 23:47), Brazil.xslt (1 KB, Apr 7 23:58), India.xslt (1 KB, Apr 8 00:04), Korea.xslt (1 KB, Apr 8 00:04), Mexico.xslt (1 KB, Apr 8 00:04), and Pakistan.xslt (1 KB, Apr 8 00:04). The 'Pakistan.xslt' file is highlighted.

Name	Size	Date
main.php	615 B	Apr 7 23:23
results.xml	1004 B	Apr 7 23:46
control.php	944 B	Apr 7 23:47
main.xslt	2 KB	Apr 7 23:47
Brazil.xslt	1 KB	Apr 7 23:58
India.xslt	1 KB	Apr 8 00:04
Korea.xslt	1 KB	Apr 8 00:04
Mexico.xslt	1 KB	Apr 8 00:04
Pakistan.xslt	1 KB	Apr 8 00:04

## 3.2. PHP Source Code

### 3.2.1 main.php

```
1 <?php
2
3 ini_set('display_errors', '1');
4 error_reporting (E_ALL);
5
6 $country = "null";
7 if (isset($_REQUEST['country'])) {
8     $country = $_REQUEST['country'];
9 }
10
11 $xp = new XsltProcessor();
12
13 $xsl = new DomDocument;
14 $xsl->load('http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/main.xslt');
15
16 $xp->importStylesheet($xsl);
17
18
19 $xml_doc = new DomDocument;
20 $xml_doc->load('results.xml');
21
22 $xp->setParameter(NULL, 'country', $country);
23
24 if ($html = $xp->transformToXML($xml_doc)) {
25     echo $html;
26 } else {
27     echo "-- could not transform --";
28     trigger_error('XSL transformation failed.', E_USER_ERROR);
29 }
30 ?>
```

### 3.2.2 control.php

```
1 <?php
2 ini_set('display_errors', '1');
3 error_reporting (E_ALL);
4
5 $country = $_REQUEST['country'];
6
7 $xp = new XsltProcessor();
8
9 $xsl = new DomDocument;
10
11 switch ($country)
12 {
13 ▼ case 1:
14     $xsl->load('http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/Brazil.xslt');
15     break;
16 ▼ case 2:
17     $xsl->load('http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/Mexico.xslt');
18     break;
19 ▼ case 3:
20     $xsl->load('http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/Pakistan.xslt');
21     break;
22 ▼ case 4:
23     $xsl->load('http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/India.xslt');
24     break;
25 ▼ case 5:
26     $xsl->load('http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/Korea.xslt');
27     break;
28
29 ▼ }
30
31 $xp->importStylesheet($xsl);
32 $xml_doc = new DomDocument;
33 ▼ $xml_doc->load('results.xml');
34
35     if ($html = $xp->transformToXML($xml_doc)) {
36         echo $html;
37     } else {
38         echo "-- could not transform --";
39         trigger_error('XSL transformation failed.', E_USER_ERROR);
40     }
41 ?>
```

## 3.3. XSLT Files

### 3.3.1 main.xslt

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
4
5 <xsl:template match="/">
6 <xsl:text disable-output-escaping='yes'><!--!DOCTYPE vxml SYSTEM "http://www.w3.org/TR/voicexml120/vxml.dtd"></xsl:text>
7
8 <vxml version = "2.0" xmlns='http://www.w3.org/2001/vxml'
9 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
10 xsi:schemaLocation='http://www.w3.org/2001/vxml
11 http://www.w3.org/TR/voicexml120/vxml.xsd'>
12 <form id="getRequest">
13 <field name="country">
14 <prompt>
15 <break/>
16 Hello! My name is rush! <break/> This is my XML project, depends on the previous result of XQuery, now I get 5 target
17 countries <break/> Please choose one country then I will give back the information about its government, overall gdp,
18 inflation and its land area per capita.
19 <break/>
20 <enumerate>
21 For <value expr="_prompt"/>
22 press
23 <value expr="_dtmf"/> <break/></break>
24 </enumerate>
25 </prompt>
26 <option dtmf="1" value="1">Brazil</option>
27 <option dtmf="2" value="2">Mexico</option>
28 <option dtmf="3" value="3">Pakistan</option>
29 <option dtmf="4" value="4">India</option>
30 <option dtmf="5" value="5">Korea</option>
31 <noinput>Sorry I did not hear any input or receive any d t m f from your phone.</noinput>
32 <nomatch>Sorry, I did not understand that country you choiced, please try again.</nomatch>
33
34 </field>
35 <block>
36 <prompt>
37 Good job! Your have picked the <value expr="country"/>.
38 </prompt>
39 <submit next="http://lyle.smu.edu/~ruixuanz/cse7347/rush/project/control.php" namelist="country"/>
40 </block>
41 </form>
42 </vxml>
43 </xsl:template>
44 </xsl:stylesheet>
```

### 3.3.2 Five Secondary Xslt Files

#### 3.3.2.1 India.xslt

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
4
5 <xsl:param name="style">India</xsl:param>
6 <xsl:template match="/">
7 <xsl:text disable-output-escaping='yes'>&lt;!DOCTYPE vxml SYSTEM "http://www.w3
8 .org/TR/voicexml120/vxml.dtd"&gt;</xsl:text>
9
10 <vxml version = "2.0" xmlns='http://www.w3.org/2001/vxml'
11 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
12 xsi:schemaLocation='http://www.w3.org/2001/vxm
13 http://www.w3.org/TR/voicexml120/vxml.xsd'>
14
15 <form><block> <prompt>
16 <xsl:apply-templates select="/myData/country/name[text()='India']"/>
17 </prompt></block></form>
18
19 </vxml>
20
21 <xsl:template match="/myData/country/name[text()='India']">
22 Oh yeah! I found the information about <xsl:value-of select="."/>. Let's
23 take a look! <break></break>
24 <xsl:value-of select="../name"/> <break></break>
25 The inflation of this country is <xsl:value-of select="../inflation"/>
26 percentage<break></break>
27 The government of this country is <xsl:value-of select="../government"/><
28 break></break>
29 The overall gdp of this country is <xsl:value-of select="../gdp"/>million
30 dollars<break></break>
31 The land area per capita of <xsl:value-of select="."/> is <xsl:value-of
32 select="../lapc"/>square meters<break></break>
33 Thank you for your phone call, Have a nice day!
34 </xsl:template>
35
36 </xsl:stylesheet>
```



### 3.3.2.2 India.xslt

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
4
5 <xsl:param name="style">Korea</xsl:param>
6 <xsl:template match="/">
7 <xsl:text disable-output-escaping='yes'>&lt;!DOCTYPE vxml SYSTEM "http://www.w3
8 .org/TR/voicexml120/vxml.dtd"&gt;</xsl:text>
9
10 <vxml version = "2.0" xmlns='http://www.w3.org/2001/vxml'
11 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
12 xsi:schemaLocation='http://www.w3.org/2001/vxm
13 http://www.w3.org/TR/voicexml120/vxml.xsd'>
14
15 <form><block> <prompt>
16 <xsl:apply-templates select="/myData/country/name[text()='Korea']"/>
17 </prompt></block></form>
18
19 </vxml>
20
21 <xsl:template match="/myData/country/name[text()='Korea']">
22 Oh yeah! I found the information about <xsl:value-of select="."/>. Let's
23 take a look! <break></break>
24 <xsl:value-of select="../name"/> <break></break>
25 The inflation of this country is <xsl:value-of select="../inflation"/>
26 percentage<break></break>
27 The government of this country is <xsl:value-of select="../government"/><
28 break></break>
29 The overall gdp of this country is <xsl:value-of select="../gdp"/>million
30 dollars<break></break>
31 The land area per capita of <xsl:value-of select="."/> is <xsl:value-of
32 select="../lapc"/>square meters<break></break>
33 Thank you for your phone call, Have a nice day!
34 </xsl:template>
35
36 </xsl:stylesheet>
```

### 3.3.2.3 Brazil.xslt

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
4
5 <xsl:param name="style">Brazil</xsl:param>
6 <xsl:template match="/">
7 <xsl:text disable-output-escaping='yes'>&lt;!DOCTYPE vxml SYSTEM "http://www.w3
8 .org/TR/voicexml120/vxml.dtd"&gt;</xsl:text>
9
10 <vxml version = "2.0" xmlns='http://www.w3.org/2001/vxml'
11 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
12 xsi:schemaLocation='http://www.w3.org/2001/vxm
13 http://www.w3.org/TR/voicexml120/vxml.xsd'>
14
15 <form><block> <prompt>
16 <xsl:apply-templates select="/myData/country/name[text()='Brazil']"/>
17 </prompt></block></form>
18
19 </vxml>
20 </xsl:template>
21
22 <xsl:template match="/myData/country/name[text()='Brazil']">
23 Oh yeah! I found the information about <xsl:value-of select="."/>. Let's
24 take a look! <break></break>
25 <xsl:value-of select="..name"/> <break></break>
26 The inflation of this country is <xsl:value-of select="..inflation"/>
27 percentage<break></break>
28 The government of this country is <xsl:value-of select="..government"/><
29 break></break>
30 The overall gdp of this country is <xsl:value-of select="..gdp"/>million
31 dollars<break></break>
32 The land area per capita of <xsl:value-of select="."/> is <xsl:value-of
33 select="..lapc"/>square meters<break></break>
34 Thank you for your phone call, Have a nice day!
35 </xsl:template>
36
37 </xsl:stylesheet>
```

### 3.3.2.4 Mexico.xslt

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
4
5 <xsl:param name="style">Mexico</xsl:param>
6 <xsl:template match="/">
7 <xsl:text disable-output-escaping='yes'>&lt;!DOCTYPE vxml SYSTEM "http://www.w3
.org/TR/voicexml120/vxml.dtd"&gt;</xsl:text>
8
9 <vxml version = "2.0" xmlns='http://www.w3.org/2001/vxml'
10 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
11 xsi:schemaLocation='http://www.w3.org/2001/vxm
12 http://www.w3.org/TR/voicexml120/vxml.xsd'>
13
14 <form><block> <prompt>
15 <xsl:apply-templates select="/myData/country/name[text()='Mexico']"/>
16 </prompt></block></form>
17
18 </vxml>
19 </xsl:template>
20
21 <xsl:template match="/myData/country/name[text()='Mexico']">
22 Oh yeah! I found the information about <xsl:value-of select="."/>. Let's
23 take a look! <break></break>
24 <xsl:value-of select="../name"/> <break></break>
25 The inflation of this country is <xsl:value-of select="../inflation"/>
26 percentage<break></break>
27 The government of this country is <xsl:value-of select="../government"/><
28 break></break>
29 The overall gdp of this country is <xsl:value-of select="../gdp"/>million
30 dollars<break></break>
31 The land area per capita of <xsl:value-of select="."/> is <xsl:value-of
select="../lapc"/>square meters<break></break>
Thank you for your phone call, Have a nice day!
</xsl:template>
</xsl:stylesheet>
```

### 3.3.2.5 Pakistan.xslt

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
4
5 <xsl:param name="style">Pakistan</xsl:param>
6 <xsl:template match="/">
7 <xsl:text disable-output-escaping='yes'>&lt;!DOCTYPE vxml SYSTEM "http://www.w3
8 .org/TR/voicexml120/vxml.dtd"&gt;</xsl:text>
9
10 <vxml version = "2.0" xmlns='http://www.w3.org/2001/vxml'
11 xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
12 xsi:schemaLocation='http://www.w3.org/2001/vxm
13 http://www.w3.org/TR/voicexml120/vxml.xsd'>
14
15 <form><block> <prompt>
16 <xsl:apply-templates select="/myData/country/name[text()='Pakistan']"/>
17 </prompt></block></form>
18
19 </vxml>
20
21 <xsl:template match="/myData/country/name[text()='Pakistan']">
22 Oh yeah! I found the information about <xsl:value-of select="."/>. Let's
23 take a look! <break></break>
24 <xsl:value-of select="../name"/> <break></break>
25 The inflation of this country is <xsl:value-of select="../inflation"/>
26 percentage<break></break>
27 The government of this country is <xsl:value-of select="../government"/><
28 break></break>
29 The overall gdp of this country is <xsl:value-of select="../gdp"/>million
30 dollars<break></break>
31 The land area per capita of <xsl:value-of select="."/> is <xsl:value-of
32 select="../lapc"/>square meters<break></break>
33 Thank you for your phone call, Have a nice day!
34 </xsl:template>
35
36 </xsl:stylesheet>
```

## 4. Conclusion

After executing every part of this project, we can get every expected result from each step. Tests have revealed this application is bug-free.

During the process of this project, I have integrated several useful XML technologies together to create an interesting application, and get an in-depth understanding of how to use XML to parse and query the data. This would help me a lot in the future learning of data analysis.