SelectChildrenAndAttributesFeature

Select Children And Attributes Feature

This feature enables dynamic selection of XmlObject properties based on their QNames. Also the QNameSet parameter methods could be used, in many cases, to select the elements and attributes that are in the any or anyAttribute wildcard buckets.

The dynamic selection is done using the following XmlObject methods:

```
    XmlObject[] selectChildren(QName elementName);

    XmlObject[] selectChildren(String elementUri, String elementLocalName);

    XmlObject[] selectChildren(QNameSet elementNameSet);

    XmlObject selectAttribute(QName attributeName);

    XmlObject selectAttribute(String attributeUri, String attributeLocalName);

    XmlObject[] selectAttributes(QNameSet attributeNameSet);
```

They select the contents of the children elements or attributes with the given name or their name is contained in the given QNameSet.

These are two useful methods on SchemaType to find out the QNameSets of elements or attributes in the wildcard buckets for a given type.

```
    public QNameSet qnameSetForWildcardElements();

    public QNameSet qnameSetForWildcardAttributes();
```

Note: The schema specification contains "Element Declarations Consistent" rule, which requires that any definition for <foo> be of exactly the same type even if it's pulled in via a wildcard (http://lists.w3.org/Archives/Public/www-xml-schema-comments/2003OctDec/0029.html). Because of this rule, these two methods will return the qname sets coresponding to the wildcards but without containing the names of the elements or attributes explicitly declared in the SchemaType or it's base types.

Example

Example user code for this feature. See below the schema and the xml instance.

Code

```
        DocDocument document = DocDocument.Factory.parse(xml);
        DocDocument.Doc doc = document.getDoc();
        Collection errors = new ArrayList();
        System.out.println(xml + "\n\nvalid: " + doc.validate(new XmlOptions().setErrorListener(errors)));
        printErrors(errors);

        XmlObject xo;
        XmlObject[] xos;

        // select a known element
        xos = doc.selectChildren(new QName(uri, "int"));
        print("1 selectChildren 'int' : ", xos);

        xos = doc.selectChildren(uri, "string");
        print("2 selectChildren 'string' : ", xos);

        // elemA
        xos = doc.selectChildren(new QName(uri, "elemA"));
        print("3 selectChildren 'elemA' : ", xos);

        // select a known attribute
        xo = xos[0].selectAttribute(new QName("", "price"));
        print("4     selectAttribute 'price' : ", xo);

        // select all attributes
        QNameSet qns = QNameSet.forWildcardNamespaceString("##any", uri);
        xos = xos[0].selectAttributes(qns);
        print("5     selectAttributes set'##any' :", xos);

        // elemB
        xos = doc.selectChildren(new QName(uri, "elemB"));
        print("6 selectChildren 'elemB' : ", xos);
        print("7     selectChildren set'##other' : " , xos[0].selectChildren(QNameSet.forWildcardNamespaceString
("##other", uri)));
        print("8     selectAttributes set'##other' : ", xos[0].selectAttributes(QNameSet.
forWildcardNamespaceString("##other", uri)));

        // elemC
        xos = doc.selectChildren(new QName(uri, "elemC"));
        print("9 selectChildren 'elemC' : ", xos);
        print("10    selectChildren set'##any' : " , xos[0].selectChildren(QNameSet.forWildcardNamespaceString
("##any", uri)));

        // select elements in the any buchet by excluding the the known elements
        QNameSetBuilder qnsb = new QNameSetBuilder();
        qnsb.add(new QName(uri, "someElement"));
        qnsb.add(new QName(uri, "aditionalElement"));
        qnsb.invert();

        print("11a    selectChildren in the any buchet for typeExtendedC: " , xos[0].selectChildren(qnsb.
toQNameSet()));

        print("11b    selectChildren in the any buchet for typeExtendedC: " , xos[0].selectChildren
(TypeExtendedC.type.qnameSetForWildcardElements()));

        // select attributes in the any buchet by excluding the the known elements
        qnsb = new QNameSetBuilder();
        qnsb.add(new QName("", "att1"));
        qnsb.add(new QName("", "aditionalAtt"));
        qnsb.add(new QName(XSI_URI, "type"));
        qnsb.invert();

        print("12a    selectChildren in the any buchet for typeExtendedC: " , xos[0].selectAttributes(qnsb.
toQNameSet()));
        print("12b    selectChildren in the any buchet for typeExtendedC: " , xos[0].selectAttributes
(TypeExtendedC.type.qnameSetForWildcardAttributes()));
```

Schema

```
<xs:schema
   xmlns:xs='http://www.w3.org/2001/XMLSchema'
   xmlns='http://xml.apache.org/test/wildcardutil'
   targetNamespace='http://xml.apache.org/test/wildcardutil'
   elementFormDefault='qualified'>

  <xs:element name='doc'>
    <xs:complexType>
      <xs:sequence>
        <xs:choice minOccurs='0' maxOccurs='unbounded'>
          <xs:element name='int' type='xs:int' />
          <xs:element name='string' type='xs:string' />
          <xs:element name='elemA' type='typeA' />
          <xs:element name='elemB' type='typeB' />
          <xs:element name='elemC' type='typeC' />
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="typeA">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs='0' maxOccurs='unbounded' />
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax" />
  </xs:complexType>

  <xs:complexType name="typeB">
    <xs:sequence>
      <xs:element name="someElement" type='xs:int' />

      <xs:any namespace="##other" processContents="lax" minOccurs='0' maxOccurs='unbounded' />
    </xs:sequence>
        <xs:attribute name='att1' type='xs:string' />

    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>

  <xs:complexType name="typeC">
    <xs:sequence>
      <xs:element name="someElement" type='xs:string' />

      <xs:any namespace="##other" processContents="lax" minOccurs='0' maxOccurs='unbounded'  />
    </xs:sequence>
    <xs:attribute name='att1' type='xs:string' />

    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>

  <xs:complexType name="typeExtendedC">
    <xs:complexContent>
      <xs:extension base='typeC'>
        <xs:sequence>
          <xs:element name="aditionalElement" type='xs:string' />
        </xs:sequence>
        <xs:attribute name='aditionalAtt' type='xs:string' />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name='topLevelElement' type='xs:string' />

  <xs:attribute name='price' type='xs:float' />
  <xs:attribute name='quant' type='xs:byte' />

</xs:schema>
```

XML Instance

```
<doc xmlns='http://xml.apache.org/test/wildcardutil'>
  <int>7</int>
  <string> ... some text ... </string>
  <elemA price='4.321'>
    <topLevelElement> this is wildcard buchet </topLevelElement>
  </elemA>
  <elemB xmlns:p='uri:other_namespace'
       p:att='attribute in #other namespace'>
    <someElement>2</someElement>
    <p:otherElement> element in #other namespace </p:otherElement>
  </elemB>
  <elemC xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
         xmlns:p='uri_other_namespace'
         xsi:type='typeExtendedC'
         att1='attribute from typeC'
         aditionalAtt='attribute added in type extension'
         p:validAtt='attribute in any buchet' >
    <someElement> element from typeC </someElement>
    <p:validElement> element in the 'any' buchet for typeExtendedC </p:validElement>
    <aditionalElement> element from typeExtendedC </aditionalElement>
  </elemC>
</doc>
```